MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

ISSUE 4
MARCH 1980 PRICE $2.50

***** CONTENTS *****

MICRO-80 is registered for posting as a publication - CATEGORY B.

Printed by: The Shovel and Bull Printers,
312 A Unley Road, Hyde Park, SA 5061.

All enquiries should be sent to:-
MICRO-80, P.O. BOX 213, GOODWOOD SA 5034
'Phone: (08) 381 8542

MICRO-80

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

** ABOUT MICRO-80 **

EDITOR: Ian Vagg
ASSOCIATE EDITORS: Peter Hartley, Eddy Paay.

MICRO-80 is the first Australian monthly magazine devoted entirely to the Tandy TRS-80 microcomputer and the Dick Smith System 80. It is available by subscription, $24.00 for 12 months or by mail order at $2.50 per copy. A cassette containing all the programs in each month's issue is available for an additional $3.50 or a combined annual subscription to both magazine and cassette, is available for $60.00. Special bulk purchase rates are also available to computer shops etc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80 or System 80 and their peripherals. MICRO-80 is in no way connected with either the Tandy or Dick Smith organisations.

** WE WILL BUY YOUR PROGRAM **

Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your TRS-80 or System 80 to earn some extra income is included in every issue.

** CONTENT **

Each month we publish at least one applications program in Level 1 BASIC, one in Level 2 BASIC and one in DISK BASIC (or disk compatible Level 2). We also publish Utility programs in Level 2 BASIC and Machine Language. At least every second issue has an article on hardware modifications or a constructional article for a useful peripheral. In addition, we run articles on programming techniques both in Assembly Language and BASIC and we print letters to the Editor and new product reviews.

** ADVERTISING **

We accept camera ready copy for display advertising at the following rates:

- FULL PAGE (19cm. wide x 24cm. high) $50.00
- 1/2 PAGE (19cm. wide x 12cm. high) $30.00

Classified advertisements are $5.00 for up to 50 words. Advertisements must be submitted by the 23rd of each month in order to appear in the following month's issue. A Company Order or payment must be included with the advertisement.

** COPYRIGHT **

All the material published in this magazine is under copyright. That means that you must not copy it, except for your own use. This applies to photocopying the magazine itself or making copies of programs on tape or disk.

** LIABILITY **

The programs and other articles in MICRO-80 are published in good faith and we do our utmost to ensure that they function as described. However, no liability can be accepted for the failure of any program or other article to function satisfactorily or for any consequential damages arising from their use for any purpose whatsoever.

- 00000 -

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

***** EDITORIAL *****

No doubt you have already noticed this issue's improved presentation. It is another sign of the continuing success of MICRO-80, thanks to the support of you, our readers. We will continue to improve the standard of our magazine in content, appearance and presentation as rapidly as possible. We will also be making a determined effort to attract advertising over the coming months. We believe that a magazine such as MICRO-80 is incomplete unless it keeps its readers informed of relevant products available in the market place. The additional revenue from advertising will also help improve the magazine still further. You can help us by urging your friends to take out subscriptions and, whenever you reply to an advertisement, tell the supplier that you saw his product in MICRO-80.

** SYSTEM 80 **

When we launched MICRO-80 it was with the aim of supporting both Tandy's TRS-80 and Dick Smith's System 80. Unfortunately, deliveries of the System 80 have been delayed for several months although we have heard that the first machines are now due to be delivered at the beginning of April. As soon as we can get our hands on a machine, we will publish a review of it. We will also purchase our own system a the earliest opportunity so we can check out our programs on it.

To avoid having to write "the Tandy TRS-80 and System 80" time and time again throughout each issue, we will henceforth ue the generic term " '80 " to describe either or both machines.

** MICRO-80 PRODUCTS **

Over the next few months, we will have some exciting announcements to make concerning new products in both software and hardware. We have established a new business, MICRO-80 PRODUCTS, to handle them and you will see advertisements for the first items in this month's issue.

You will also see a review of the book, LEARNING LEVEL II, by David A. Lien, who wrote the USER'S MANUAL FOR LEVEL 1, supplied with their machines by Tandy. It is an excellent book and we have placed a bulk order for it with the publisher. David Lien was in Australia recently and told us that the first printing of 11000 copies had sold out already and a second printing is now underway. This means that there is likely to be a slight delay in deliveries but we will accept orders, with payment, on a first-come-first served basis.

Some time before the next issue is published, we expect to move to new premises where we will have a new telephone number, complete with an answering machine. We also expect to become an authorised BANKCARD merchant. We will then offer a 24 hour telephone order service for subscriptions and products. Look for a detailed announcement in next month's issue.

** POSTAL DELAYS **

It has come to our notice that some readers did not receive their February issues until well into March, if at all. If you already had a subscription at the beginning of February, then your February issue was posted out on 20th February. According to Australia Post, all copies should have been delivered within five working days ie. by 27th February. If you have still not received your February issue, please write to us and we will send you a replacement and make a claim on Australia Post. To help cope with this problem in future, we now stamp the date of posting on the back of each mailing sleeve. When you receive your copy, check the posting date and, if it is earlier than five working days from the date received, complain to Australia Post. Perhaps then we can speed things up.

- 00000 -

***** NEXT MONTH'S ISSUE *****

The April issue of MICRO-80 will contain at least the following programs and articles:

HORSE RACE (L1) - All the fun and excitement of a day at the races, complete with course and moving horses, and a running race call to boot. An odds on favourite in anyone's book.!!

ONE-ARM BANDIT (L1) - Las Vegas here we come! Select the size of your bet and the '80 will do the rest.. complete with tumbling wheels and even a moving handle. A real jackpot - this one.

KRAZY KAT (L2) - You control the cat. the computer contols the mouse, and there's a maze between the two. Offers eight different situations and three levels of skill. Fun for all from 7 to 70!

ONE-ARM BANDIT (L2) - This program does for Level 2 machines what the above program does for Level 1 machines. just the Authors are different. Now you can lose your money whatever system you use!

MOVIE-MAKER (L2) - If you like to write programs which employ moving graphics, then this utility program, written in BASIC, is just what you have been looking for. We guarantee that you will not find a way to make faster graphics in less program space, unless you write in machine language.

B MON - Part 3 m/c language - The final part of this great utility program This program, coupled with MOVIE-MAKER, will save you hours of programming time.

LOWER CASE MOD. - This is the same lower case mod. that we at MICRO-80 use on our own computer. It will enable you to display lower case letters and a whole bevy of new symbols on your screen and print them on your printer (if it supports lower case. As usual, the hardware is only as good as the software that drives it, so we will alo publish programs, both in BASIC and m/c language to enable you to make the most of this cheap, (< $10) modification.

The April issue will also contain the next instalment on Assembly Language Programming and all the usual features such a BETTER BYTES etc..

- 00000 -

***** ASSEMBLY LANGUAGE PROGRAMMING - Part 3 *****

by Edwin R. Paay.

** INTRODUCTION **
I have been asked to write the assembly language articles from this month on. so here goes! My aim is to help you all become assembly language programmers but to do this I will need some feedback. If there is anything in these articles which you cannot understand or if I miss something out that you want to know. please write in and ask, that way we can make sure that this series is just what you want.

** ASSEMBLY VERSUS MACHINE LANGUAGE **
Some of you are probably wondering what the difference is between assembly language and machine language (note that m/l and a/l will be the abbreviations for machine language and assembly language respectively for the rest of the articles). A m/l program is a code that can be decoded and acted upon directly by the CPU (central processing unit). It is a set of binary coded instructions stored in memory. Binary numbers are too long and cumbersome to work with so we use hexadecimal notation to represent them. That is all very well for the CPU, for the programmer however it is impossible to remember hundreds of machine language instructions in hexadecimal. This is where a/l comes in, it gives us a convenient means of writing programs.
A/l uses mnemonics which is a form of shorthond notation, for instance in machine language (also called object code) the instuctions 21 - 34 - 12 mean: load the HL reg. poir with the value 1234H (from now on, all hexadecimal numbers will have an "H" behind them to set them apart). In a/l (also called source code) this would be: LD HL,1234H as you can see, that is a lot easier to remember. Typing in "LD HL,1234H" is easy for the programmer but to the CPU it means nothing os yet, which is where the assembler comes in. It converts the a/l text (source code) into m/l (object code) which can then be loaded by the computer and executed. Now, don't think of a/l just as discrete instructions, it is a proper programming language, with features which makes it, in some respects, easier to use than BASIC.

** WHY ASSEMBLY LANGUAGE PROGRAMMING ? **
At this point I would like to make the following statement : "Assembly languoge is the most powerful programming language on any computer system." More powerful than BASIC, FORTRAN and PASCAL combined. Let me explain. Take the case of the TRS-80, for instance, it contains a basic interpreter in ROM. this is just a m/l program which can interpret BASIC statements and act on them accordingly. So we can see that m/l can do the same things that BASIC does. In the same way we could write a m/l program thot recognises FORTRAN or PASCAL statements, so m/l is at least as powerful as all these languages combined. Every programmer (this means you !) can use m/l programs or suboutines to advantage, and often it is the only way some things can be done. For instance, I needed a program to draw graphics lines rapidly between any two points on the screen, the SET command under BASIC was too slow ond POKE statements were not suitable either, so the only thing left to do was to write an a/l program to do the job for me, that is how "SET2" in last month's issue came about.

The most common reasons for using a/l programming are its high speed, complete control of the computer, more efficient memory use or to provide functions that are not available through basic.

Just before we get down to business, a word about programming aids. In Part 1 we mentioned that you will need an Editor/Assembler, a debug/monitor and a disassembler programs to make the most of this series. For the really serious a/l programmer I can also recomend the Texas Instruments "TI programmer" calculator. I personally would not like to be without it, apart from the normal calculator functions this calculator also allows conversions between octal, hexadecimal and decimal, and it will do almost all the logic and arithmetic functions of which the Z80 is capable, which can be very helpful in debugging.

The next few parts in this series of articles will deal mostly with the different types of instructions available on the Z80. When that has been done, I will prepare some articles on actual programming and reveal some useful ROM routines which can be used in your m/l programs. I hope that the information presented so far has helped some of you to get a better picture of what a/l programming is all about, if you want to become an a/l programmer then read on.

--------==== DATA MOVEMMENT====----

Of all the instructions available, the LD (load) command is the most used in almost all a/l programs. There are 8 bit (one byte) loads and 16 bit (two bytes) loads. we will handle each seperately. First let us look at some 8 bit load instructions :

```
10      LD      A,5H
20      LD      B,D
30      LD      E,(HL)
40      LD      (BC),A
50      LD      (IX+3),L
60      LD      A.(4567H)
70      LD      (9ABCH),A
```

Now we will take each one in turn :

Line 10 says - load the "A" register with 5 Hex.; so after execution of line 10, reg. A will contain 5.

Line 20 says - duplicate the value contained in the "D" reg. into the "B" reg; therefore if "D" contains 9, after execution of line 20 the "B" reg. will contain 9 also and "D" remains unchanged.

Line 30 means - load the "E" reg. with the contents of whatever address the "HL" reg. pair is pointing to.

Example: if, before execution of line 30, HL = 5000H and memory location (address) 5000H contains 12H, then after execution "E" contains 12H, "HL" and address 5000H remain unchanged.

Line 40 says - load address pointed to by the "BC" reg. pair with contents of the "A" reg.

Example : if, before execution "A" = 22H and "BC" = 2233H then, after execution memory location 2233H will contain 22H , "BC" and "A" are unchanged.

Line 50 shows the use of the index registers, this line says ; load memory location pointed to by reg. "IX" plus 3 , with the same value as is currently in the "L" reg.

Example : if "IX" contains 5555H and L = 10H then, after executing this instruction, memory location 5558H will contain 10H, "L" and "IX" reg. will remain unchanged.

Note that the value "3" in line 50 can range from +127 to -128 (decimal) this is called the displacement. the "IY" reg. functions in the same way. This function is suitable for accessing tables containing data etc.

Line 60 has a value in between the brackets, this value points directly to the location in memory to be used.
Example: if locotion 4567H contains 50H then, after execution of line 60. register "A" will contain 50H and address 4567H will remain unchanged.
Line 70 - apart from a different address between the brackets, is line 60 in reverse. It puts the value from "A" into location 9ABCH and leaves "A" unchanged.

These lines show some of the different ways in which the "LD" function can be used to transfer 8 bit bytes between main memory and the registers. At this point, I must mention that not all combinations of "LD" functions are legal, E.g.:

```
        LD      BC,L
        LD      (1234H),H
        LD      (BC),(HL)
        LD      (DE),5H
```

are all illegal, so you can see that, initially, an a/l manual is needed to see which combinations are legal. After some practice you will know which are legal and which are not. Also, you may have noticed that if a value or reg. pair is placed between brackets it means it is pointing to a location in memory, as in the case of lines 30 to 70 incl., for more info. on the addressing modes, review last month's issue. Now we will look at some 16 bit loads:

```
15      LD      HL,6023H
25      LD      DE,(5019H)
35      LD      (789AH),BC
```

These are some typical 16 bit loads.
Line 15 loads the value 6023H in the "HL" reg. pair, so after execution "H" contains 60H and "L" contains 23H. When combined, "HL" =6023H
Line 25 reads, load the value contained in location 5019H and 501AH in memory into the DE reg. pair.
Example : if 5019H = 22H and 501AH = 11 then after execution "DE" = 1122H. Note that "D" reg. = 11H and "E" =22H. The Z80 and the Intel 8080 store data in a "back to front" format; for instance, if "BC" = 3456H then, after executing line 35 "C" which contains 56H, is dumped to memory first so that memory loction 789AH will contain 56H. The Z80 will then dump the contents of the "B" register (34H) to the next location 789BH.
Line 25 showed us that it takes data from memory least significant byte first so that data will be read back from memory correctly.
It must also be mentioned that the flags are not affected by any of the data moving operations (LD,POP,PUSH,EX,etc)
There are two 16 bit instructions which are part of the 16 bit LD group but which are different from the rest:- PUSH and POP. These instructions involve the stack pointer, so we will need to discuss the stack first.

** THE STACK, PUSH and POP
The Z80 contains a 16 bit register labeled "SP" (stack pointer). The value in this reg. points to a location in memory, this location is called the stack area. The stack pointer can be set by using ; LD SP,nn ,where "nn" can be any four digit Hex. value. There will be more on the stack in a future issue.
Now to get back to our PUSH and POP instuctions, they look like this :

```
22      PUSH    HL
44      POP     BC
```

these instructions will, of course, work with any of the reg. pairs IX,IY,HL,DE,BC,AF and not just with HL and BC as shown in the lines above.
Example : if "HL" = 1234H and the "SP" contains 5525H then, after line 22 is executed, "HL" will be unchanged, the "SP" will contain 5523H, memory location 5524H will contain 34H and location 5523H will contain 12H.
Note that the stack pointer moved down to a lower value memory location and that it decremented by 1 before it pushed the first byte on the stack, and the stack pointer always points to the last byte pushed onto the stack. Now, if the CPU executes line 44 straight after line 22, the SP will contain 5525H again and the "BC" reg will contain 1234H. First notice that the stack is a "LIFO" stack ie. last in first out, in other words the last value that was PUSHED on the stack will be the first to POP off the stack when POPped.
By now I am sure some of you are starting to wonder how you will remember all this and are starting to feel a bit down-hearted, well here is some good news, with the PUSH and POP instructions, all that is really neccessary to remember is that the order in which you PUSH registers on to the stack must be the reverse of the order in which you POP them off again later. I have gone into detail here so that you can refer to it when needed, to find out exactly how the PUSH and POP functions work.
The stack can be used for temporary storage of data or for saving data from the registers so that the registers can be used by subroutines (e.g. when doing a X=USR(0) routine from BASIC), then, upon returning from the subroutine, the original data can be restored to the registers by POPping them off the stack. A second important use for the stack is as a medium when transferring data from one 16 bit reg. to another. This is what was done in lines 22 and 44 above. After executing lines 22 and 44 there has simply been a transfer of data from the "HL" reg. pair to the "BC" reg. pair.

** THE EXCHANGE INSTRUCTIONS **

There is a group of instructions called "EX"changes, I will list a few below:

```
13      EXX
33      EX      DE,HL
```

The instruction in line 13 exchanges data in BC,DE and HL with their equivalents in the alternate reg. set, this is the only way the alternate registers can be accesed for use.(see last month's issue for disscusion on Z80 reg. set)
Example : if HL = 10H, DE = 12H, BC = 55H, HL' = 43H, DE' =49H and BC' = 99H, then, after execution of line 13 HL = 43H, DE = 49H, BC = 99H, HL' = 10H, DE' = 12H, BC' = 55H, (note 55H= 0055H , suppressed leading zero's to make numbers shorter), so it can be seen that EXX has simply swapped 3 reg. pairs with the equivalent reg. pairs in the alternate reg. set.
Now we get to line 33 - if "HL" = 6474H and "DE" = 0897H, then after execution of line 33 "HL" = 0897H and "DE" = 6474H, so it has simply swapped the data in them, around.

** BLOCK TRANSFERS **

Now we come to the LDIR instruction. This is a very powerful function and is not found on any other 8-bit microprocessor that I know of, it looks like this:

```
83      LDIR
```

This instruction will move a block of memory up to 64k bytes long to any location in memory. This would normally be quite a lang subroutine on any other processor, on the Z80 however it is a single instruction. To use it we must load the DE,HL and BC reg.'s with the relevant data. The "HL" reg. must contain the source address, "DE" reg. must contain the destination address and "BC" reg. must be loaded with the number of bytes to be moved.
Example : if we wanted to move 3 bytes starting at 5000h to 8000h in memory a suitable program would look like this :

```
11      LD      HL,5000H
12      LD      DE.8000H
13      LD      BC.3H
14      LDIR
```

If locations 5000H = 11H, 5001H =21H and 5002H =31H then after execution of the program above 5000H to 5002H are unchanged "BC" will contain zero (the LDIR command uses the BC reg. as a byte counter and keeps on looping until BC=0) "HL" reg. pair will contain 5003H, "DE" will contain 8003H. The addresses 8000H to 8002H will then have the following data in them ; 8000H=11H, 8001H=21H, 8002H=31H.
There is also an LDDR instruction, this does the same as the LDIR instruction, except that it decrements the "DE" and "HL" reg. pairs. What this means is that the LDIR starts at the top of a block of memory (lowest address first) and the LDDR at the bottom (highest address first), the LDDR instruction is not as often used as the LDDR instruction.

I will now list a small sample program for you to experiment with using some of the functions discussed in this article :

```
10                  ORG     4F00H
20      LOOP        LD      HL,3C00H
30                  LD      DE.3C01H
40                  LD      BC.400H
50                  LD      (HL).A
60                  INC     A
70                  LDIR
80                  PUSH    AF
90                  LD      BC.3FFFH
100                 CALL    60H
110                 POP     AF
120                 JR      LOOP
130                 END     LOOP
```

I will go through this one line at a time.
Line 10 simply tells the assembler that the program will be located starting at 4F00H (to allow for 4k system users).
Line 20 first of all has a label called "LOOP", this is so we can loop back to it later, it then tells the CPU to load the "HL" reg. pair with 3C00H, this is the start of video memory.
Line 30 loads the "DE" reg. pair with 3C01H which is the second byte in video memory.
Line 40 loads the "BC" reg. pair with 400H, this is the number of bytes in the block of video memory.
We now have the registers set up for a block move.

Line 50 loads the first byte of video memory (because line 20 loaded "HL" with 3C00H) with the contents of the "A" reg. we don't know at this time what the "A" reg. contains. It really doesn't matter because we are going to let the program cycle right through the complete ASCII set anyway, so we don't care where we start.
Line 60 simply increments the value in the A" reg. by one. The "A" reg. will now contain the ASCII code for the next character, which is used when it loops back and executes line 50 again.
Line 70 does a block move, the blocks in this case however overlap and the result is that, whatever character is in the first byte of video memory, is duplicated in the whole block of video memory. In other words, it fills the screen with whatever was in the first video location on the screen.
Line 80 saves the contents of the "AF" reg. set on the stack. This is necessary because we are going to execute a subroutine which will destrou the value in the "AF" reg. set and it is up to us to save it.
Line 90 loads the "BC" reg. pair with a value, this value will be used in a delay loop. The delay will be 3FFFH times 14 micro-seconds, or approx. 0.25 second.
Line 100 directs the computer to the subroutine stored in ROM located at 60H. The "CALL" instruction has the same function as the "GOSUB" statement in BASIC. 60H is the start of the delay routine I was talking about above.
Line 110 restores the data pushed earlier to the "AF" reg. set.
Line 120 causes the program to loop back to line 20 and go through the whole sequence again.
Line 130 tells the assembler it is the end of the program and that the entry point (start) of the program is at the line called "LOOP" (line 20) .
There are a few instructions in the program that have not been discussed so far, this will be done in one of the future articles.

The program above will fill the screen with the complete ASCII character set, including graphics. As you will notice, m/l graphics are fast, the characters will appear to fill the screen immediately. In fact, this program is so fast that if we didn't put in the delay loop the scanning of the screen would not be able to keep up with the computer. You can try this by taking out the delay loop.
For those of you who do not have an assembler I will list a basic program which will load the m/l program for you using POKE statements :

```
10 DATA 33,0,60,17,1,60,1,0,4,119,60,237,176,245
20 DATA 1,255,63,20S,96,0,241.24,233
30 FOR X=20224 to 20247: READ A : POKEX,A :NEXT X
40 POKE 16525,240: POKE16527,79: X=USR(0)
50 END
```

To disable the delay loop change line 20 to:

```
20 DATA 1,255,63,0,0,0.241,24,233
```

To be continued.

- 00000 -

***** ABBREVIATED ABBREVIATIONS - By Charlie Bartlett *****

For all you frustrated owners of Level 1 4K machines who are trying to fit in 5K of program, here is something that may help. Below is a list of all the abbreviations for Level 1. Some of them are listed in the Level 1 owners manual but some are not. So even if you think you know them all, it will still pay you to look through the list.

ABS(X)	=A.(X)	INT(X)	=I.(X)	RETURN	= RET.
CLS	=C.	NEXT	=N.	RESET(X,Y)	=R.(X,Y)
DATA	=D.	ON	=O.	RND(X)	=R.(X)
END	=E.	PRINT	=P.	STOP	=ST.
FOR	=F.	PRINT AT	=P.A.	STEP X	=S.X
GOTO	=G.	POINT(X,Y)	=P.(X,Y)	SET(X,Y)	=S.(X,Y)
GOSUB	=GOS.	RESTORE	=REST.	TAB(X)	=T.(X)
INPUT	=I.	READ	=REA.	THEN	=T.

In addition to using the above abbreviations here are some other tips which will help you save space:-

(1) Do not use REM lines.
(2) Do not put blank spaces anywhere that you don't have to.
(3) Use multiple statement lines whenever possible,
(4) Do not repeat program lines, if you need it more than once, put it in a subroutine.
(5) Do not use arrays ("A(2)=7") for variables unless you must.
(6) Leave out instructions where possible or keep them short.

Happy squeezing!

***** LEVEL 1 PROGRAM SELECTOR - by Charlie Bartlett *****

If you have several programs loaded onto one tape and you don't want the first one, type in this line in the COMMAND MODE (no line numbers) and go and have a cup of coffee. When you return, the tape will be in position to CLOAD the program you want in the normal manner.

```
        F.X=1 TO (N-1):P."PROGRAM";X;"LOADING":IN.#A$:N.X
```

NOTE: N=the number of the program you want, if the one you want is the third on the tape, for example, you substitute 3 for N.

- 00000 -

***** BETTER BYTES *****

a potpourri of ideas, hints, tips, and odd ramblings, based on readers' letters, and compiled by Peter Hartley.

DAMAGE can result from connecting your power supply to the wrong socket. Mark the power cord clearly by fixing a strip of sandpaper - sand outwards - round the plug, with a rubber band. This way, if your don't see it, you will feel it before it's too late!

RECORD/PLAY head in a floppy drive is in actual physical contact with the disk during read/write operations. Hence the need to protect from greasy fingers, falling hair, fag ash, etc. Almost makes those plastic covers seem worth-while!

MULTIPLE dumps are a good idea if you cannot afford TANDY data tapes. This line gives you three dumps with gaps between, while you wander off and tell the wife how brilliant you've been!

```
FOR T=0 TO 1500: OUT 255,4: NEXT: FOR K=1 TO 3:CSAVE"N": FOR T=0 TO 1500: OUT
255,4: NEXT: NEXT
```

EXPECT to see some hefty drops in the retail price of Floppy Drives soon if our information is correct. Of course, the poor old Aussies will be the last to benefit!

USEFUL routine for drawing a box round the '80's screen using only one FOR/NEXT loop.

```
100 FOR VAR = 0 TO 127
110 SET (VAR,0): SET (VAR,127)
120 IF VAR>47 SET (0,VAR): SET (47,VAR)
130 NEXT
```

NOTICE that you-know-who's latest newsletter is described as "Volume No. 7". I've missed out on everything since "Volume 1, Issue 6"!

READERS contributions are invited for this column.

***** READER'S REQUESTS *****

This column is a regular feature of MICRO-80. In it. we list all those articles, programs etc., requested by our readers. We then do our best to work our way through them in the coming months. If you are thinking of contributing an article or program, have a look at this list first, it will give you an idea of what our readers want.

** ARTICLES **

File handling on the TRS-80
Book reviews
Review of printers for the TRS-80
Reviews of commercially available software and hardware
Review of Chess programs
Description of the functions performed by the Expansion Interface

** SOFTWARE **

GAME OF LIFE relocated to start at 7000H
RICOCHET for L2/4K machines
M/c language program to use BREAK key like RESET when Expansion Interface is connected.
PRINT#-1,INPUT#-1 speedup
Hex to Dec, Dec to Hex conversion routines

** HARDWARE **

RS232 printer Interface
Interfacing the TRS-80 to external hardware
Lowercase modification

- 00000 -

***** A REVIEW OF "LEARNING LEVEL II" - By Charlie Bartlett *****

(Charlie has just converted his Level 1 machine to Level II. We thought he was the ideal person to review this American book which has been receiving rave reviews in the USA).

I have just read a copy of "Learning Level II" written by David A. Lien who also wrote the Level 1 Manual. He has adopted a similar approach and the result is a Level II Manual which is as easy to read and understand as the Level 1 book and just as much fun.

The first chapter in the book is devoted to updating the Level 1 Manual to read like a Level II. For someone starting out with a Level II machine and having no previous experience with the BASIC language, this section takes care of an area not previously covered by other books. The whole of the first chapter is reproduced at the back of the book so that, should you wish to cut out the amendments and stick them in the Level 1 Manual, you can do so without damaging the main part of the book. Chapters 2 and 3 deal much more fully with the EDITOR than the official manual does and would give anybody a much better understanding of its abilities.

Chapters 4 to 22 deal with the other Level II commands, written in the same clear, operator error trapping, style that most of you know from Level 1. Chapters 23 to 25 deal with the expansion interface, dual cassette operation and real time clock, also very easy to follow (must be I don't have one but after reading it I know how to use one). The Appendix contains a very much expanded explanation of the Error codes an ASCII code table, various useful information and the reproduction of Chapter 1 as mentioned above.

Well, those are all the good things. The bad? The ONLY thing that seems to be missing is any reference at all to the STRING$ function. Apart from that it is an excellent book which I would recommend to anybody and should be included by Tandy with their Level II machines but, considering the likelihood of that, I suggest you buy one yourself. Those of you who already have Level II. "well, you see there are these functions and routines not mentioned in the official Manual." What are they? I'd tell you, but I have run out of room!

- 00000 -

***** SOFTWARE SECTION *****

The listings for these programs are at the back of the magazine. This section explains what each program is about and how to use it. Level 1 programs have been printed by first converting them to Level 2. When entering them, use abbreviations and remember to enter "PRINT @" as "PRINT AT".

** LEARNING NIM LI/4K ** by Marlan Binet

Nim is one of many names given to this game. where two players - in this case you and the '80 - take turns to remove not less than one nor more than three objects from a pile. The loser is the player who removes the last object. Sounds easy?

This version offers three modes of play...
a) where you can win most of the time
b) where you might win occasionally and
c) where the '80 learns from its mistakes and eventually becomes unbeatable.

It is this third mode that demonstrates a simple form of artificial intelligence. By keeping a record of those play positions which have always resulted in a win, the '80 gets progressively better in its play until it is truly unbeatable.

Just enter the program, type RUN and take it from there.

** LUNAR LANDER - L1/4K **

This program is presented in two, 4k modules. The first contains the instructions, the second is the program itself. All that needs to be said here is good luck, you are certainly going to need it.

** RICOCHET - L2/4K **

This is the program, first published in the January issue, now rewritten to run on Level 2 machines. The special feature of the Level 1 program was the way in which it simulated the INKEY function. It did this by SETting two graphics blocks which were subsequently overwritten by an alpha character when one of the control keys was operated. So, why didn't it work with Level 2? The POINT statement was used to interrogate the graphics blocks to decide if they were still SET or not. A SET block returns the value +1 in Level 1 but -1 in Level 2, hence the problem. However, this version of the program uses the INKEY function, so no problems.

The program commences by asking you to specify a level of play between easy (1) and difficult (10). It then draws a wall down the centre of the screen. Randomly spaced along the wall are some gaps, the number of which is determined by the degree of difficulty chosen - 6 gaps for easy, 15 gaps for difficult. Lined up with these gaps is an equal number of targets to the far right. Walking up and down the screen to the left of the wall is a man. When you think he is in the right position, press the "F" key, the man will stop, raise his arm and fire. If you stopped him correctly, the bullet will pass through the gap in the wall and hit the target. If not, the bullet will ricochet off the wall and kill him! You have 3 men targets per game. When the man reaches a boundary, he automatically reverses, unlike the Level 1 game, you cannot reverse his direcion from the keyboard. Sounds easy doesn't it - try it!!!

** INVADERS (L2/16K) - by Ron Sully **

The game INVADERS is built essentially around the Level 2 INKEY$ function and PEEK and POKE statements. The INKEY$ function is used to fire and move the cannon and PEEK and POKE are used in moving and firing of the INVADERS. The RND function and plenty of GOSUBs and GOTOs help to create the illusion of many things happening at once.

Apart from initially setting the stage, there is no true start point as the program passes randomly from and to the following actions:

1. The movement of the cannon
2. The firing of the cannon
3. The movement of the INVADERS
4. The firing of the INVADERS
5. The firing of the static INVADERS (Mother Ships)

as well as scoring and creating simulated explosions where necessary.

The game was designed to be competitive. That is one player would play to determine a score then each consecutive player would try to better that score. But it is also suitable as a solitaire game in that any one player can endeavour to better his own score with each consecutive game. A game can last between five and six minutes and scores can exceed 6000.

** RANDOM NUMBER GENERATOR TEST PROGRAM (L2/16K) - by Keith Neighbour **

This program draws a table showing the frequency of occurrence of numbers between 1 and (up to) 105 selected by the TRS-80 as random numbers by the command RND. You can allow the sample to run on indefinitely or you can limit it by requesting a specific number of randomized samples.

A table of the range of numbers chosen for sampling (1 to whatever) is laid out on the screen bounded by > and < in each line (program lines 60 to 82). Hence limitation to 105 as the highest number, as 1 to 105 gives a screenfull.

The frequency of the random choice of each number is displayed beneath its number, and this display is constantly updated.

As each random number is chosen it is briefly displayed at the bottom right hand corner of the screen together with the frequency count for that number and the total samples to date.

The progressive random number cycling routine shuttles between program lines 90 and 160. After generating a random number in line 90, subroutine 500 chooses the correct frequency display line to be updated (the busy *s show which line is being worked on) and Line 110 chooses the correct over-print position in that particular frequency display line.

The display can be interrupted at any time and restarted by hitting any key. If you interrupt the run and then hit "L" the number(s) with the highest and lowest frequencies are displayed in the format :

```
               HIGHEST: 7 (times) FOR 34,26
               LOWEST : (etc).....
```

The number of times that each number is chosen (its frequency) is stored in array C ( ). (see line 100)

Subroutines 1000 to 1090 and 1100 to 1190 search through array C( ) and identify the highest and lowest freguencies for the appropriate numbers.

A screen print subroutine is used (lines 20000 to 20100) for hard copy if required. This is activated by hitting P during an interrupt.

If you chose a high range of numbers for random sampling - say between 1-50 and 1-105, let the program run for a minute or two before interrupting to find the numbers with the highest and lowest frequency, otherwise the list of numbers with lowest freguency (zero in this case) will swamp the screen!

Of course the longer the program is run the more accurate the sampling will be and the more even the distribution will be (assuming the random number generator is a good one!). To try out any random number generator for good freguency distribution, replace the statement R=RND(E) in Line 90 with the particular random number generator routine. A good random number generator will give an equal frequency for each number.

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

** RANDOM NUMBER GENERATOR TEST PROGRAM NO.2 (L2/16K) - by Keith Neighbour **
This is similar in operation to the previous program except that the results are shown in a constantly up-dated bar graph form. The range of numbers to b randomised has also been limited to the numbers 1 to 10.
The secret to the fast graphics display is the direction of the bar graphs which run horizontally rather than vertically, as is usual. The length of th bar graph against the horizontal scale indicates the number of times (th frequency) each number from 1 to 10 has been chosen by the random numbe generator. The exact frequency is displayed to the left of each of th numbers 1 to 10 on the vertical scale.
The up-dating is continuous and the graphic display re-generates itself whe the horizontal scale bacomes too small to accomodate the "leading" number. The horizontal scale is then automatically increased by a factor of 10 and th bar graphs are shortened to their correct length for the new scale.
Line 260 prints the total numberr of random numbers to date at the bottom o the screen. Line 265 prints the frequency on the left of each of numbers 1 t 10. Line 270 converts the frequency to the number of graphic blocks for th particular bar graph. Line 280 prints the bar graph.
The cycle runs between lines 200 and 310. Line 235 decides when a new grap scale is needed and sends the program to subrotine 4000 which draws the nex generation of bar graphs.
Routine 3120 to 3170 prints the vertical display scale and routine 3180 t 3200 prints the horizontal display scale.
Interrupt and resume operation by hitting any key.

** ANALOGUE CLOCK (L2/16K) - by Keith Neighbour and Peter Hartley **
This program draws a very realistic clock face on the screen, then draws an redraws the hour, minute and second hands very rapidly indeed, using the SET program published last month. Unless you have that progrram, this clock wil not work.
Type in the BASIC program listing, RUN it to check that it draws the cloc face satisfactorily, (you will get an error message in line 650) then CSAV it.
To load the complete program, you need to have put SET2 on tape. Power-up answer MEMORY SIZE ? with 32222, then type SYSTEM, load in SET2 and type "ENTER". You now have the SET< > command in your machine. The next step i to CLOAD the BASIC clock program you CSAVEd earlier, RUN it and you arre i business.
If the clock is inaccurate, adjust the timing loop in line 540. Increase th value to more than 515 if the clock is runnig fast, decrease it if it i running slow. Incidentally, this program also features a silent chime on th hour. You might not be able to hear it but it sure shakes the clock about!!

** BMON PART2 (L2/16K) - by Eddy Paay **
THIS MONTHS INSTALMENT WILL ENABLE THE RENUMBER, LOAD, MERGE, PROTECT AN CANCEL PROTECTION COMMANDS. ONE THING I DIDN'T MAKE CLEAR LAST MONTH IS THA THE MEMORY SIZE HAS TO BE SET TO 29216 ON POWER UP.

RENUMBER.
THIS ROUTINE WILL RENUMBER BASIC PROGRAMS TO USER SUPPLIED PARAMETERS. IF BASIC PROGRAM IS IN MEMORY AND THE RENUMBER COMMAND IS CALLED UP THE FIRS THING YOU WILL SEE IS A BLOCK OF GRAPHICS FLASH IN THE LOWER RIGHT CORNER O THE SCREEN. THIS TELLS YOU THAT THE PROGRAM IS BEING PREPARED FOR RENUMBERING IT WILL THEN ASK "I?"; ANSWER THIS WITH THE INCREMENT FROM LINE TO LINE, THE IT WILL ASK "N?"; ANSWER THIS WITH THE LOWEST NUMBER LINE THAT YOU WANT. TH INDICATOR WILL THEN FLASH AGAIN TELLING YOU RENUMBERING IS IN PROCESS. WHE

FINISHED THE COMMAND LIST WILL BE DISPLAYED TO THE SCREEN AGAIN AND IT IS POSSIBLE TO RETURN TO BASIC.

LOAD.
THIS COMMAND WILL LOAD BASIC PROGRAMS FROM TAPE AND DISPLAYS THEM TO THE SCREEN WHILE LOADING. AFTER TYPING THE "L" KEY THE MESSAGE "READY CASSETTE" WILL FLASH ON THE SCREEN, THEN HOLD DOWN THE ENTER KEY UNTILL THE RECORDER STARTS TO RUN. THE BREAK KEY CAN BE USED TO ABORT LOADING BUT ONLY WHILE DATA IS BEING READ, NOT FOR INSTANCE WHILE IT IS READING THE LEADER.
WHEN IT IS FINISHED LOADING THE END OF THE PROGRAM WILL BE MARKED WITH A LARGE BLOCK OF GRAPHICS, AND THE PROGRAM LOCATIONS IN MEMORY ARE SHOWN. (NOTE THAT IT DOES NOT DISPLAY THE LINE NUMBERS)

MERGE.
WHEN MERGING USE THE SAME PROCEDURE AS FOR "LOAD" IT WILL THEN LOAD A NEW PROGRAM BEHIND ONE ALREADY IN MEMORY. WHEN IT IS THROUGH LOADING IT WILL DISPLAY A BLOCK OF GRAPHICS IN THE RIGHT LOWER CORNER. THIS MEANS IT IS AUTOMATICALLY RENUMERING THE NEW PART OF THE PROGRAM TO MAKE IT FIT BELOW THE OLD ONE. WHEN THE INDICATOR STOPS FLASHING IT SIGNIFIES THAT RENUMBERING IS FINISHED YOU CAN THEN HIT ANY KEY TO RETURN TO THE DIRECTORY.
NOTE THAT IF THE BREAK KEY IS PRESSED WHEN THE "READY CASSETTE" MESSAGE APPEARS INSTEAD OF THE ENTER KEY, THE CANCEL PROTECTION COMMAND (THE "F" KEY) MUST BE USED TO RECOVER ANY PROGRAMS IN MEMORY.

PROTECT PROGRAM AND CANCEL PROTECTION.
THIS COMMAND WILL PROTECT A BASIC PROGRAM AND MAKE IT APPEAR AS IF IT IS NOT THERE. IT CAN THEN BE RECALLED USING THE CANCEL PROTECTION COMMAND LATER.

- 00000 -

***** HARDWARE SECTION *****

** DOUBLE THE STORAGE CAPACITY OF YOUR DISKS **

The standard disk drive, of the type supplied by Tandy, uses only one side of each disk. However, the disks themselves have magnetic material on each side and drives such as the PERCOM, sold by Dick Smith, use both sides. How do they do it? If you look at a disk with the label towards you in the bottom left hand corner, you will see one notch and two holes in its cover (not including the large centre hole). The rectangular WRITE PROTECT NOTCH is at the top of the disk, directly above the right hand edge of the label. As the disk is inserted into the drive with this notch uppermost, the edge of the disk deflects an arm attached to a micro switch. But, when the disk is pushed right in, the arm is so positioned that it drops into the notch and changes the state of the micro switch. This is sensed as a non write-protected disk. You can WRITE PROTECT the disk by putting a piece of strong tape over the notch. This prevents the arm from dropping into the notch the microswitch does not change state and the drive interprets the disk as being WRITE PROTECTED. Drives such as the PERCOM have a second microswitch and arm assembly at the bottom of the drive so that a disk can be sensed for write protection when put in either way up.

Now, if you look at the disk cover again, you will see a small round hole. close to the centre hole at about "1 o'clock". This is the index/sector hole. If you CAREFULLY rotate the disk inside its cover by gripping it with your fingers at the extreme inside edge of the large centre hole, you will find that there is a much smaller hole in the disk which lines up with the index/sector hole, once per revolution. The Tandy disk drive has a LED on one side of the disk and a detector on the other side, lined up with the index/sector hole. Once each revolution, the small hole in the disk lets a pulse of light from the LED through to the detector. This is used as a synchronisation signal so that the drive knows where the disk segments start. The PERCOM disk drive has this same arrangement but, in addition, it has a second LED and detector positioned at about "5 o'clock" with respect to the disk. Thus, when you turn the disk over in a Percom drive, this second LED and detector will be aligned with the index/sector hole and, in conjunction with the wright protect micro switch at the bottom of the drive, enables the reverse side of the disk to be used in exactly the same way as the normal side.

Now, the installation of a second write protect switch and LED/detector into an existing Tandy disk drive, to make it double-sided, would be a formidable task, unless you happen to have a tame machine shop handy. There is, however, another very simple way to make a Tandy drive "double-sided". It involves modifying the disk covers themselves. Even if we cannot put a write protect swith at the bottom of the drive and an index/sector LED/detector at 5 o'clock, at least we can put a write-protect NOTCH in the bottom of the disk cover and a second index/sector hole in the cover at 5 o'clock. Then, when we turn the disk over, the drive will find all the signals it needs and will happily write and read to the reverse side of the disk.

** The Template **
Towards the back of the magazine, you will find a template which looks just like the reverse side of a disk cover. That's exactly how we made it! You will find it has the two essential requirements, a write/protect notch at the bottom and an index/sector hole at "5 o'clock". Cut this out and glue it to a piece of stiff cardboard which you should also cut to the exact outline of the template, including the write/protect notch. You now have a solid cardboard replica of a disk cover. By one means or another, you need to make a hole right through the cardboard and the template, where the index/register hole is on the template. You could use a 1/4 inch wad punch, or a drill or a nail and razor blade to trim the material flat.

So far you havn't had to spend very much but now comes the expensive bit. You need a one-hole paper punch (1/4 inch). These usually resemble a pair of pliers. You must make sure that the punch you use will cut a clean hole through the material that disk covers are made from. We use a punch made in England by MAUN INDUSTRIES - Part No. 4603-160. This punch is a high quality unit but it is not cheap. Ours cost $9.25. We have seen other one-hole punches for much less. The local K-MART for instance, had one at 93 cents the other day. Whether these are good enough for the job we cannot say but, for 93 cents you might like to try one. If you do buy the Maun Industries tool, you will need to remove the finger screw and paper stop from the lower jaw.

The only other equipment you need is a medium sized white TEXTA marker pen and a small rectangle of thin cardboard about 2 1/2 cm. wide by 5 cm. long with the two corners at one end slightly bevelled to remove sharp edges (we cut a piece out of the pack the Tandy disks came in).

Now to modify your first disk. Take your time and don't rush things, it only takes a few minutes anyway. Place the disk to be modified on a table with the write/protect notch at the top and the label in the bottom left hand corner

Place your template over it so that the write/protect notch in the template is on the bottom edge of the disk and the two elongated slots through which the disk head reads and writes, line-up at "3 o'clock" (you could cut this slot out in your template too, if you like). Now, carefully adjust the position of the template so that all its edges are alligned with those of the disk. At this point, if you look straight down on the template, you should not be able to see the disk below it except through the write/protect notch and the index/sector hole. Now, take the white Texta marker and mark through the template onto the disk cover, the shapes of the write/protect notch and the index/sector hole. Remove the template. You should now have a disk with a white rectangle on its bottom edge, almost touching the right hand side of the label and a white circle close to the large centre hole at about "5 o'clock". Take the one-hole punch and punch a semi-circular write/protect notch so as to cut out the white mark on the bottom edge of the disk. This notch works just as well as a semi-circle it does not have to be rectangular. Now, take the small piece of cardboard and slip the bevelled end in between the disk and the cover, under the white mark for the index/sector hole. This cardboard is to prevent the punch from damaging the disk surface. It is important that the cardboard should be clean and should not crumble or leave any fibres behind. Carefully lift up the cover and slip the bottom anvil of the one-hole punch between the cover and the piece of cardboard. Take care to allign the punch over the white spot and punch it out. Inside the black disk cover there is a specially treated inner cover of white tissue paper. Make sure that the punch has cut this out too. If it has not cut quite cleanly, carefully pull the partially cut circle out towards the centre of the disk until it comes away. DO NOT LET ANYTHING GET OR REMAIN INSIDE THE DISK COVER. Remove the punch and piece of cardboard.

Place the template on top of the disk, as you did before then slip the template behind the disk (as though you were shuffling cards). Carefully align all the edges again and, keeping them in alignment, turn both disk and template over. You should now be looking at the reverse side of the template. Take your white marker and mark the index/sector hole again. Repeat the procedure with the piece of cardboard and one-hole punch as you punch the index/sector hole in the other side of the cover. Remove the cardboard and punch and you are finished. We usually use the white marker to write a large "B" on the reverse side and an "A" on the label side of the disk cover.

Disks modified this way will work equally well on either side. You may write-protect either side by putting a piece of tape over the appropriate write/protect notch. Incidentally, these modified disks WILL NOT work on a Percom drive unless you block off the new index/sector hole on one side. This is best done with a small piece of black cardboard over the hole, held in place with tape. (Do not put the tape directly over the hole, it might stick to the disk).

Now, see how long it takes you to fill up those hundreds of freeby grans you have just discovered!!!

- 00000 -

***** LETTERS TO THE EDITOR *****

From: K.L. ELLIS, Randwick, NSW.

May I initially take this opportunity to congratulate you and your staff o the quality and usefulness of MICRO-80. I can only hope that it will continu to flourish. It provides a medium that is long overdue in the TRS-80 world. I disassembled Eddy's BMON-Pt.1 program out of interest and was delighted t see the use that had been made of entry points and routines within the ROM Quite some time ago I wrote a pogram which disassembled the whole of the Leve 2 ROM and disk basic RAM in the hope of gaining access to the routines etc This was reasonably successful but, as you will be well aware, it is a mammot task and unless one has nothing else to do with one's life, you don't get ver far. What would be extremely useful to me and I have no doubt to a great man other assembler programmers, would be a description of the ROM routines, thei entry points and parameters, for both Levels 1 and 2. I am sure that amongs all those dabbling in this area, there would be a considerable amount o valuable information but have not as yet, seen a printed collection of thi material. I sincerely hope that the staff and readers of MICRO-80 will b able to do something in this area in the not too distant future.

Peter Hartley's article on the Light Pen was extremely useful. I have bee using a Light Pen of similar design for some time now and, as Peter has found it suffers from all sorts of peculiarities. I wrote an assembler program t allow the user to draw on the screen and the BIT instruction solved many o the problems inherent in the CASSIN data latch. The fact that the Video RA is multiplexed between CPU and Video chain divider causes some problems wit the high speed of the machine code, but a reasonable delay loop fo multiplexing can compensate well for this design problem. Light Pe sensitivity and problems associated with the automatic volume control of th cassette recorder, remain the last two hurdles to be overcome. I a considering a simple port interface direct to the system bus, thus eliminatin the cassette recorder altogether. I would be very interested to hear fro readers who have been experimenting with Light Pens.

I was extremely pleased to read your comments regarding educational progra for student use. For far too long this area has been hampered by cheap, nast programs. All due praise to MICRO-80 for making a stand!

In conclusion may I suggest that in order to enhance an already magnifice magazine, a half-yearly or yearly cross-referenced index be produced. would be extremely useful in collating all the myriad of useful bits a pieces contained in each issue.

(We are presently preparing a book containing a description of the R routines and how to use them. We expect to have it available towards the e of April but will make an announcement closer to the time.

We would be interested in seeing the disassembler you wrote and also t assembly language program for the Light Pen. Why not send them in so we c consider them for publication? Thanks for the suggestion about t cross-referenced index, we will publish one at the appropriate time. -Ed.)

From: B.H. CHRISTENSEN, Woden A.C.T.

I would like to express my appreciation that someone IN AUSTRALIA has tak the initiative to publish a magazine like MICRO-80. I have been extreme pleased with the first three issues and I hope you keep up the good work. am especially appreciative of your balance between hardware and software as am interested in both. The following are my suggestions for future topics.

On the hardware side, you have indicated that a number of projects are in the pipeline eg. memory expansion. Already on the market there are a number of devices designed for systems without the expansion interface. For example, I own a printer interface (Tandy type) and a Morse Code-RTTY interface. What I really am going to need is some kind of matherboard so that these and other devices can all be plugged in at once. I guess you have this sort of thing in mind but I would ask that you publish the interface standards (S100 or whatever you choose) early so that other constructors would have a standard to work to. For my part, I am working on a "real" real-time clock that will be independent of the TRS-80 but able to be interrogated by it to provide date and time without the need to reset on powering up. If it ever works I will send you the details.
On the software side, I think the thing I would like most is a BASIC Compiler designed so that a program can be written and debugged in interpreter mode using the ROM and then compiled and run as a machine language program for speed and low memory usage. I am also looking forward to the disassembly of the ROM and the various ROM routines that you promised.

(Thanks for the encouragement. We do have a motherboard in mind and will get around to it shortly, after the memory board is finished. As far as the BUS is concerned, we are keeping to the Tandy standard. We have considered S100 but it really does seem rather pointless and expensive, to convert the Tandy BUS to S100 then decode from S100 on each plug-in board. The memory board, for example, would become much more expensive since it would have to provide its own refresh signals as well. That is not to say that we will ignore S100 in the future since the System 80 will use it.
We would be interested in seeing the results of your hardware real-time clock with a view to publishing it. The software clock which comes with TRS DOS has proved to be more of a liability than an asset to most users and we notice that its accuracy (importance?) has been downgraded in TRS DOS 2.3.
We agree that a BASIC compiler would be nice. We are only aware of two suitable programs, both American and both costing in the vicinity of $350! If any of our readers has written a compiler of their own, please let us know - Ed.)

From: JOHN C. HARDWICKE, Collaroy Plateau, NSW.
I would appreciate some advice through your columns, on the most economical way to include diskette and printer on my TRS-80 Level 2/16K machine. I am not interested in the narrow paper strip printer, rather a unit capable of at least quarto size with ribbon printing.
Is information on your lower case mod available?

(There is a lot of interest in printers John and we will shortly start researching the whole subject for a future article. The lower case mod will be published next month. -Ed)

From: G. STEWART, Glenroy Vic.
I am not able to make the program, MERGE (MICRO-80 Dec 79) work in the way in which it was intended. When MERGE is the only program in memory, I get the given addresses from the PEEK statements. Upon CLOADing a BASIC program, the LIST reveals that, not only has MERGE been erased from memory but so have th first 10 lines of the BASIC program. This happens both after typing NEW and after power-off. Advice on how to overcome the difficulty would be appreciated and equally importantly, an explanation in terms of machine operation for the seeming malfunction, is sought.

If a reader has written a m/c language program that speeds up data transfer during PRINT#-1 and INPUT#-1 operations, would they please offer it for publication in MICRO-80?

(We are as puzzled as you by this problem. On receiving your letter, we loaded MERGE from the December issue cassette but it worked perfectly for us. Are you sure you are following the instructions exactly? If it still won't work in your machine, perhaps you would be better to forget it and use BMON, which does everything MERGE does, but better. Perhaps some reader has encountered this fault condition and can write in giving us an explanation?
A program to speed up PRINT#-1 and INPUT#-1 would be useful, we have put it in our Readers' Requests column. Ed)

From: RON KEHN, Karumburra, Vic.
I would like to express my thanks to Ronald J. Sully for fixingup the errors in the Basic Monitor listed in Issue 2. Like Romald, I turned my attention to the Life game. Using the monitor, I was eventually able to get the machine code loaded and the program to run. The procedure I used was as follows:

1. Power on and protect memory at 31800
2. CLOAD the corrected monitor and use it to create the hex listing of LIFE, starting at 31888 (7C90H) - arbitrary choice.
3. Use the monitor to dump the HEX code to tape - I used the parameters 31888,32360,32206.
4. Next, I deleted all lines under 8000 thus leaving only the LOAD portion. I inserted the following line:
   8036 START=20480:FINISH=20952
   then I CSAVEd this special program, calling it Loadlife.
5. Next, I powered-off then powered-on again, protecting memory at 20479.
6. I then CLOADed Loadlife - ran it to load the HEX dump created previously and, with line 8036, it gets put in the correct location in memory.
7. Finally, I CLOADed the BASIC part of LIFE, typed in RUN and sat back to enjoy the intriguing patterns the computer produced.

The above procedure was carried out on a Level 2, 16K machine.

(Thank you for this useful contribution, Ron. Loading the machine language portion of LIFE is even easier now, using the MINI MACHINE LANGUAGE SAVE and LOAD programs published in February's issue but the principle is essentially the same as the one you have described - Ed.)

From: Mr. A. L' ESTRANGE, Darwin,NT.
(Mr. L' Estrange explains how he debugged Monitor in Basic, then continues -)
Could you provide a subroutine that would scroll a program listing one line at a time and possibly any designated number of lines at a time?
I had previously written Hex to Dec and Dec to Hex routines but the routines in Monitor in Basic were beautiful, I would never have thought of that method.

(We have put your request for the LISTing program into our Readers' Requests column - it should be possible to write such a program in machine language. Thanks for the nice comments about the Hex to Dec and Dec to Hex routines. Your interest has prompted us to prepare an article on this subject for a future issue. - Ed.)

From: Mr. F. G. STURGES, Townsville, Qld.
As your Z80 BASIC is a little different from mine, is there an available book giving a function listing?
Eg. your "PRINT @ A5,A$" is unintelligble to me,
as is "IF (A(A(0)))>0*(A(A(0)))<27"

(In order to use progams written for the '80. on a different machine, you would need to fully understand Level 1&2 BASICs plus, of course, the BASIC on your own machine. Level 1 in a "TINY BASIC" taking 4K of ROM whilst Level 2 is an extended BASIC taking 12K of ROM. The book, Learning Level II, by David A. Lien, which you will find reviewed in this issue, is a excellent way to learn all about Tandy's BASIC. There is another book by the same author - The Basic Handbook, which is an encyclopoedia of the BASIC language which you could also find very useful in converting programs. I believe this book could be available from The Technical Book and Magazine Co. Pty. Ltd., 295-299 Swanston St., Melbourne 3000, Ph. (03) 663 3951. If not, we could arrange to import copies for you and any other interested readers. When imported in small quantities, the book would cost about $20.00, including air mail postage.
Undoubtopopedly the most difficult part of converting programs from one system to another is the conversion of screen formatting and graphics. The Tandy screen can display 1024 characters at once. Each character position is individually addressable via the PRINT @ statement (PRINT AT in Level 1). Position 0 is in the top left hand corner of the screen, position 63 is at the top right hand corner, position 64 is immediately below position 0 and so on until position 1023, which is in the bottom right hand corner of the screen. The PRINT @ statement will work with a constant, Eg. PRINT @ 500, or with a variable, Eg. PRINT @ A5, (where A5 has previously been given a numerical value) or will even evaluate an algebraic expression itself, Eg. PRINT @ (5+64*I).
The other expression you mention involves the logical AND operation, represented by the * and the ( ) around the expressions in the IF THEN statement. It is further complicated by the use of a "double" Array function. A(0) is the value of an Array variable. A(A(0)) is the value of another Array variable occupying the position in the array determined by the value of A(0). The expression quoted is saying:
IF the value of A(A(0)) is greater than 0, AND if it is less than 27 THEN......
ie. If the value of A(A(0)) is between 0 and 27, then the conditional statement is satisfied and program control passes to the "THEN" statement.
I hope these explanations have helped clarify the specific case you mention, Mr. Sturges. I am sure you will find the books recommended above would make conversion much easier. - Ed.)

- 00000 -

***** PROGRAM LISTINGS *****

** LEARNING NIM **

```
5  FORI=0TO59:A(I)=RND(3):NEXTI
10  CLS:FOR J=1 TO 5
20  FOR I=1 TO 100:NEXT I
30  PRINT AT 468,"WELCOME TO NIM"
40  FOR I=1 TO 100:NEXT I
50  CLS:NEXT J
60  INPUT"DO YOU REQUIRE INSTRUCTIONS(1=YES,2=NO)";Q
65  IF(Q<>1)*(Q<>2)THEN 60
70  IF Q=1 THEN GOSUB 10000
80  PRINT"HOW MANY MATCHES DO YOU WANT"
90  PRINT"IN THE PILE(ENTER 0 FOR RANDOM AMOUNT)";
100  INPUT N
105  IF(N<0)+(N>30)THEN 80
110  IFN=0THEN N=RND(20)+10
130  INPUT"SKILL LEVEL(1=FAIR,2=GOOD,3=LEARNING)";S
140  IF(S<1)+(S>3) THEN 130
150  CLS:FORI=0TO127:SET(I,25):NEXTI:PRINT@537;" MATCHES";
155  GOSUB 1000
160  PRINT@128;"THERE ARE . .";N;" MATCHES IN THE PILE"
180  PRINT@576;:INPUT"DO YOU WANT TO GO FIRST(1=YES,2=NO)";Q
190  IFQ=2GOTO2000
200  PRINT@576;:PRINT:INPUT"HOW MANY MATCHES DO YOU WISH TO TAKE";Q
210  IF(Q<1)+(Q>3)+(Q>N)THEN200
215  IFS=3THENA(N+29)=Q
220  GOSUB 3000
230  GOTO 2000
1000  FORI=3TON*3STEP3
1010   FORJ=13TO21
1020    SET(I,J)
1030   NEXT J
1040  NEXT I
1050  RETURN
2000  IFN=1THEN30000
2010  IFN=0THEN32000
2020  IFS=1THENGOSUB4000:IFRND(3)=1THENQ=RND(3)
2030  IFS=2THENGOSUB4000
2040  IFS=3THENQ=A(N-1)
2045  IF(Q>=N)+(Q=0)THENQ=ABS(Q-1):GOTO2045
2050  PRINT@640;"I WILL TAKE";Q;" MATCHES"
2060  FORI=1TO1000:NEXTI:GOSUB3000:GOTO200
3000  FORI=N*3TO(N+1-Q)*3STEP-3
3010   FORJ=13TO21
3020   RESET(I,J)
3030   NEXTJ
3040  NEXTI:N=N-Q
3050  PRINT@128;"THERE ARE . . .";N;" MATCHES IN THE PILE";:RETURN
4000  Q=INT(((N-1)/4-INT((N-.985)/4))*4):RETURN
10000  CLS:PRINT"NIM IS A GAME OF LOGICAL SKILL IN WHICH"
10010  PRINT"YOU WILL PIT YOUR WITS AGAINST THE COMPUTER."
10020  PRINT"THE GAME IS PLAYED WITH A SET AMOUNT OF MATCHES"
10030  PRINT"(YOU DECIDE THIS) AND A LIMIT ON THE AMOUNT OF MATCHES"
10040  PRINT"EACH PLAYER IS ALLOWED TO TAKE.":PRINT
10050  PRINT"THE OBJECT OF THE GAME IS TO MAKE YOUR"
10060  PRINT"OPPONENET (IN THIS CASE THE COMPUTER) PICK UP"
```

```
10070  PRINT"THE LAST MATCH EACH PLAYER TAKING TURNS"
10080  PRINT"IN TAKING FROM 1 TO 3 MATCHES FROM THE PILE."
10090  PRINT:PRINT"IN THIS GAME THE SKILL OF THE COMPUTER"
10100  PRINT"IS SET AT A CHOICE OF THREE LEVELS FAIR,GOOD OR"
10110  PRINT"LEARNING.":PRINT
10130  INPUT"PRESS 'ENTER' TO CONTINUE";A$
10140  CLS:PRINT"IN THE LEARNING MODE THE COMPUTER STARTS"
10150  PRINT"BY TAKING A RANDOM AMOUNT OF MATCHES AWAY"
10160  PRINT"BUT LEARNS FROM YOUR PLAY AND IMPROVES WITH"
10170  PRINT"EACH GAME UNTIL IT IS AS GOOD AS YOU ARE."
10180  PRINT:PRINT"THE NUMBER OF MATCHES IN THE PILE"
10190  PRINT"IS SET BY YOU TO A VALUE OF BETWEEEN 1 AND"
10200  PRINT"30 OR IS SET AT A RANDOM VALUE BY THE"
10210  PRINT"COMPUTER IF YOU SO DESIRE.WELL THATS"
10220  PRINT"ABOUT ALL YOU NEED TO KNOW."
10230  PRINT"G O O D L U C K ! ! ! !"
10240  INPUT"PRESS 'ENTER' WHEN READY";A$
10250  CLS:RETURN
30000  IFS=3THENFORI=0TO29:A(I)=A(I+30):NEXTI
30005  FORI=1TO1000:NEXTI
30010  CLS:PRINT"WELL DONE YOU HAVE WON THE GAME"
30020  IFS=3PRINT"BUT I WILL PLAY BETTER NEXT TIME"
30030  GOTO32500
32000  CLS:PRINT"I HAVE WON THE GAME"
32500  INPUT"CARE TO PLAY AGAIN(1=YES,2=NO)";Q
32510  IFO=1THENCLS:GOTO80
32520  PRINT"SEE YOU ROUND"
```

- 00000 -

** LUNAR LANDER INSTRUCTIONS **

```
1  H=5000:V=0:M=0:T=0:U=0:G=0:R=0:N=0
2 A=0:P=0
10 CLS
20 GOSUB6000
90 PRINT
100 PRINT"          READ ON FOR SOME DETAILED INSTRUCTIONS------------↑"
120 PRINT
140 GOSUB7000
150 CLS
160 PRINT"PLACE YOUR LEFT HAND ON THE KEY MARKED < A > "
170 PRINT"PLACE YOUR RIGHT HAND ON THE < ENTER > KEY "
180 PRINT
190 PRINT"GOOD, NOW EVERY TIME YOU PRESS THE < A > KEY YOU WILL COMMAND"
200 PRINT"THE ON BOARD COMPUTER TO INCREASE THE FUEL RATE, WHICH IT WILL"
210 PRINT"THEN DISPLAY ON THE CONTROL BOARD....  SO FAR SO GOOD."
220 PRINT"      NOW, WHEN YOU PRESS THE < ENTER > KEY, IT WILL COMMAND"
230 PRINT"THE ON BOARD COMPUTER TO FIRE THE PROPULSION UNIT AT THE RATE"
240 PRINT"PROGRAMMED BY YOU.  *** NOW PAY ATTENTION... IT WILL KEEP"
250 PRINT"FIRING THE MOTOR AT THAT RATE UNTIL YOU PRESS THE < A > KEY "
260 PRINT"AGAIN. ONCE IT IS PRESSED A SECOND TIME, THE MOTOR WILL STOP"
270 PRINT"AND THE FUEL RATE WILL RETURN TO ITS LOWEST SETTING."
280 INPUT"PRESS ENTER FOR A PREVIEW OF THE CONTROL BOARD";A$
300 GOSUB2000
320 PRINT@654,"***REMEMBER***";
```

```
340 PRINT@832,"LEFT HAND SELECTS : RIGHT HAND FIRES : LEFT HAND AGAIN"
350 PRINT@896,"STOPS MOTOR AND RESTORES FUEL RATE VALUE TO LOW."
360 PRINT@960,
370 GOSUB7000
390 CLS
400 PRINT"IF YOU THINK YOU UNDER STAND PRESS ENTER."
405 PRINT"IF NOT PRESS < BREAK > AND GO OVER IT AGAIN";
410 INPUTA$
420 CLS
430 PRINT"TYPE NEW AND THEN CLOAD TO LOAD IN REAL TIME LUNAR LANDER."
450 END
2000 CLS
2010 FORY=2TO9
2015 SET(48,Y):SET(49,Y)
2016 SET(66,Y):SET(67,Y)
2020 NEXTY
2030 FORX=48TO65
2035 SET(X,2):SET(X,9)
2040 NEXTX
2050 PRINT@89,H;
2060 PRINT@153,"HEIGHT";
2070 FORX=25TO45
2075 SET(X,11):SET(X,18)
2080 NEXTX
2085 FORY=11TO18
2090 SET(24,Y):SET(25,Y):SET(46,Y):SET(47,Y)
2095 NEXTY
2100 PRINT@269,M;
2105 PRINT@333,"FUEL";
2110 FORX=68TO91
2115 SET(X,11):SET(X,18)
2120 NEXTX
2125 FORY=11TO18
2130 SET(68,Y):SET(69,Y):SET(90,Y):SET(91,Y)
2135 NEXTY
2140 PRINT@291,-V;
2145 PRINT@355,"VELOCITY";
2150 PRINT@452,P;
2155 PRINT@516,"FUEL RATE";
2156 PRINT@530,"DURATION";
2157 PRINT@466,U;
2160 FORX=6TO107
2165 SET(X,20):SET(X,27)
2170 NEXTX
2175 FORY=20TO27
2180 SET(6,Y):SET(7,Y)
2185 SET(106,Y):SET(107,Y)
2186 SET(34,Y):SET(35,Y):SET(58,Y):SET(59,Y)
2190 NEXTY
2195 FORX=6TO107
2200 SET(X,29):SET(X,36)
2205 NEXTX
2210 FORY=29TO36
2215 SET(6,Y):SET(7,Y)
2220 SET(106,Y):SET(107,Y)
2225 NEXTY
```

```
2230 PRINT@644,"MESSAGE";
2235 PRINT@708,"BOARD   ";
2255 FORY=38TO45
2260 PRINT@544,"TIME";
2265 FORY=20TO27
2270 SET(78,Y):SET(79,Y)
2275 NEXTY
2280 PRINT@552,"G FORCE";
2285 PRINT@480,T;
2290 PRINT@488,A;
2350 PRINT@832," ";
3000 RETURN
6000 CLS
6005 PRINT" REAL TIME MOON LANDING.  < I N S T R U C T I O N S   O N L Y >
6010 PRINT
6015 PRINT"THIS SIMULATION IS IN REAL TIME. IT TAKES 60 SECONDS TO RUN"
6020 PRINT"FROM START TO FINISH."
6030 PRINT"IF YOU DO NOT THINK YOU CAN MANAGE IT IN THAT TIME"
6035 PRINT"TYPE 2. THIS WILL SLOW THE GAME DOWN TO HALF SPEED."
6040 PRINT"SO THAT THE 60 SECONDS DISPLAYED WILL ACTUALLY HAVE TAKEN"
6050 PRINT"120 SECONDS OF REAL TIME. ALL TIMES STATED ARE OF COURSE"
6060 PRINT"PRESUMING THAT YOU FAIL TO REDUCE SPEED AT ALL."
6095 RETURN
7000 INPUT"PRESS ENTER TO TURN THE PAGE";A$:RETURN
```

- 00000 -

** LUNAR LANDER **
(NOTE: WHEN ENTERING THIS PROGRAM, TYPE P.AT..., INSTEAD OF PRINT@...;)

```
1 GOSUB6000
2 W=0:I=0:R=0:Z=0:P=0:B=4:C=40:D=1:F=43:U=0:J=0
10 T=0:V=-25:E=200:G=2:H=5000:M=25000
11 GOSUB2000
12 T=0:V=-25:E=200:G=2:H=5000:M=25000
14 SET(B,C):SET(D,F)
15 GOSUB100:IFR<500GOTO25
20 PRINT@654;"FUEL RATE TO HIGH";:PRINT@718;"MOTOR BURNT OU
T";:GOTO50
25 GOSUB200:IFA<13GOTO35
30 PRINT@654;"G FORCES TOO HIGH";:PRINT@718;"LANDER BREAKING UP";:GOTO75
35  IF M<=5000GOTO45
36 N=N-1:IFH<=0GOTO57
40 IFN<=0GOTO15
41 GOTO25
45 PRINT@654;"NO FUEL LEFT";
50 PRINT@718;"PREPARE FOR LANDING";:R=0
51 PRINT@452;R;
55 GOSUB200
56  IF H>0GOTO55
57 IFV>=0 V=-V
60 PRINT@832;"TOUCH DOWN AT";T;"S TERMINAL SPEED=";-V;"M/S";
65 IF V>-1GOTO85
66 IFV>-5GOTO90
67 IF V>-10GOTO95
```

```
70 PRINT@896;"YOU MADE A HOLE ";M*V*V/5ØØØØ;"M DEEP."
71 FORX=1TO2ØØØ:NEXTX
75 PRINT@832;"DO YOU WANT TO PLAY AGAIN ? TYPE 1 FOR YES, 2 FOR NO";
77 INPUT" ";Q
79 IFQ=2GOTO999
80 IFQ=1GOTO2
82 GOTO75
85 PRINT@654;"GOOD LANDING";:GOTO75
90 PRINT@654;"ROUGH LANDING";:GOTO75
95 PRINT@654;"LANDER DESTROYED";:GOTO75
1ØØ H=INT(H):M=INT(M):V=INT(V):A=INT(A)
1Ø1 PRINT@89;H;:PRINT@291;-V;:PRINT@269;M-5ØØØ;:PRINT@48Ø;T;:PRINT@4Ø8;A;
1Ø2 PRINT@466;J;:PRINT@832;" ";:IFL=2THEN1Ø9
1Ø7 FORX=1TO26Ø:NEXTX:GOTO11Ø
1Ø9 FORX=1TO55Ø:NEXTX
11Ø IF(POINT(B,C)=Ø)*(POINT(D,F)=1)GOSUB4ØØØ
111 IFR>=ØTHEN115
112 GOSUB12Ø:GOTO11Ø
115 N=1:IF(POINT(B,C)=Ø)*(POINT(D,F)=Ø)GOSUB5ØØØ
116 N=INT(N):IFN>ØGOTO118
117 GOSUB12Ø:GOTO115
118 RETURN
12Ø PRINT@7Ø8;"IMPOSSIBLE";
2ØØ M=M-R:A=E*R/M-G:T=T+1:H=H+V+A/2:V=V+A
2Ø1 IFR>ØJ=J+1
2Ø2 IFR=ØJ=Ø
2Ø3 PRINT@654;"                        ";
2Ø4 PRINT@718;"                        ";
2Ø5 IFJ>ØTHENPRINT@654;"MAIN MOTOR IGNITION";
2Ø6 IFJ>ØTHENPRINT@718;"SEQUENCE NOW ON     ";
2Ø7 IFJ=ØTHENPRINT@654;"INITIATING MAIN MOTOR";
2Ø8 IFJ=ØTHENPRINT@718;"SHUTDOWN NOW         ";
3ØØ RETURN
999 CLS
1ØØØ PRINT"PROGRAM TERMINATED"
1Ø1Ø PRINT"END"
1Ø2Ø END
2ØØØ CLS:FORY=2TO9:SET(48,Y):SET(49,Y):SET(66,Y):SET(67,Y):NEXTY
2Ø3Ø FORX=48TO65:SET(X,2):SET(X,9):NEXTX:PRINT@89;H;:PRINT@153;"HEIGHT";
2Ø7Ø FORX=25TO45:SET(X,11):SET(X,18):NEXTX:FORY=11TO18:SET(24.Y):SET(25.Y)
2Ø95 SET(46,Y):SET(47.Y):NEXTY
21ØØ PRINT@269;M-5ØØØ;:PRINT@333;"FUEL";:FORX=68TO91:SET(X,11):SET(X,18):NEXTX
2125 FORY=11TO18:SET(68,Y):SET(69,Y):SET(9Ø,Y):SET(91,Y):NEXTY
214Ø PRINT@291;-V;:PRINT@355;"VELOCITY";:PRINT@452;P;:PRINT@516;"FUEL RATE":
2156 PRINT@53Ø;"DURATION";
2157 PRINT@466;U;
216Ø FORX=6TO1Ø7
2165 SET(X,2Ø):SET(X,27)
217Ø NEXTX
2175 FORY=2ØTO27
218Ø SET(6,Y):SET(7,Y)
2185 SET(1Ø6,Y):SET(1Ø7,Y)
2186 SET(34,Y):SET(35,Y):SET(58,Y)-SET(59.Y)
219Ø NEXTY
2195 FORX=6TO1Ø7
22ØØ SET(X,29):SET(X,36)
```

```
2205 NEXTX
2210 FORY=29TO36
2215 SET(6,Y):SET(7,Y)
2220 SET(106,Y):SET(107,Y)
2225 NEXTY
2230 PRINT@644;"MESSAGE";
2235 PRINT@708;"BOARD  ";
2255 FORY=38TO45
2260 PRINT@544;"TIME";
2265 FORY=20TO27
2270 SET(78,Y):SET(79,Y)
2275 NEXTY
2280 PRINT@552;"G FORCE";
2285 PRINT@480;T;
2290 PRINT@488;A;
2350 PRINT@832;" ";
3000 RETURN
4000 REM
4005 I=I+1
4020 ONIGOSUB4100,4120,4130,4140,4150,4160,4170,4180,4190,4200,4210,4220
4025 SET(B,C):SET(D,F)
4026 PRINT@452;P;
4030 RETURN
4100 P=25
4119 GOTO4025
4120 P=50
4129 GOTO4025
4130 P=75
4139 GOTO4025
4140 P=100
4149 GOTO4025
4150 P=150
4159 GOTO4025
4160 P=200
4169 GOTO4025
4170 P=250
4179 GOTO4025
4180 P=300
4189 GOTO4025
4190 P=350
4199 GOTO4025
4200 P=400
4209 GOTO4025
4210 P=450
4214 K=556
4219 GOTO4025
4220 P=500
4235 GOTO4025
5000 R=P
5015 SET(B,C):SET(D,F)
5020 P=0
5025 I=0
5110 RETURN
6000 CLS
6005 PRINT"          REAL TIME MOON LANDING"
6010 PRINT
```

```
6015 PRINT"THIS SIMULATION IS IN REAL TIME. IT TAKES 60 SECONDS TO RUN"
6020 PRINT"FROM START TO FINISH."
6030 PRINT"IF YOU DO NOT THINK YOU CAN MANAGE IT IN THAT TIME"
6035 PRINT"TYPE 2. THIS WILL SLOW THE GAME DOWN TO HALF SPEED."
6040 PRINT"SO THAT THE 60 SECONDS DISPLAYED WILL ACTUALLY HAVE TAKEN"
6050 PRINT"120 SECONDS OF REAL TIME. ALL TIMES STATED ARE OF COURSE"
6060 PRINT"PRESUMING THAT YOU FAIL TO REDUCE SPEED AT ALL."
6090 INPUT"TYPE 1 FOR REAL TIME, 2 FOR HALF TIME";L
6095 RETURN
```

- 00000 -

** RICOCHET - L2 **

PRESS F TO FIRE. NO DIRECTION REVERSE.

```
1 CLEAR10:CLS:INPUT"PLEASE ENTER LEVEL OF PLAY REQUIRED. (1 TO 10)";MM:CLS:MM=MM
+5
5 RANDOM:A=11:B=12:C=13:D=14:E=15:F=16:G=17:H=18:I=29:J=30:K=31:L=32:M=33:N=34:O
=35:Q=2:Z=1:DD=15469:EE=15474
10 FORY=1TO14:FORX=DDTOEE:POKEX,191:NEXT:DD=DD+64:EE=EE+64:NEXT
15 GOSUB2000
24 R=RND(40):IFR<4GOTO24
26 IFNN=MMGOTO40
30 X=91:IF(POINT(X,R)=0)OR(POINT(X,R-1)=0)OR(POINT(X,R+1)=0)GOTO24
35 FORX=90TO101:RESET(X,R):SET(110,R):NEXT
37 NN=NN+1:GOTO24
40 FORX=0TO127:SET(X,0):SET(X,47):NEXT:FORY=0TO47:SET(0,Y):SET(127,Y):NEXT:PRINT
@27,"RICOCHET";
50 PRINT@978,"TARGETS HIT";TT;": MEN DEAD";SS;:SET(D,I):SET(E,I)
55 SET(A,J):SET(B,J):SET(C,J):SET(D,J):SET(E,J):SET(F,J):SET(G,J):SET(H,J):FORY=
KTOL:SET(A,Y):SET(C,Y):SET(D,Y):SET(E,Y):SET(F,Y):SET(H,Y):NEXT:FORY=MTOO:SET(C,
Y):SET(F,Y):NEXT:SET(B,O):SET(G,O)
56 S$=INKEY$:IFS$="F"GOSUB1000
60 IFZ=1GOTO96
65 RESET(F,O):RESET(G,O):SET(G,N):Z=1:IFQ=1RESET(G,O-1)
70 GOTO97
96 RESET(B,O):RESET(C,O):SET(B,N):Z=2:IFQ=1RESET(B,O-1)
97 IFO=1GOTO105
100 I=I-1:J=J-1:K=K-1:L=L-1:M=M-1:N=N-1:O=O-1:IFI=1THENQ=1
102 GOTO120
105 I=I+1:J=J+1:K=K+1:L=L+1:M=M+1:N=N+1:O=O+1:IFO=46THENQ=2
107 GOTO132
120 RESET(A,L+1):RESET(B,J+1):RESET(B,O+1):RESET(C,O+1):RESET(D,L+1):RESET(E,L+1
):RESET(F,O+1):RESET(G,O+1):RESET(G,J+1):RESET(H,L+1):GOTO150
132 RESET(A,J-1):RESET(B,J-1):RESET(C,J-1):RESET(D,I-1):RESET(E,I-1):RESET(F,J-1
)
150 RESET(G,J-1):RESET(H,J-1):GOTO50
1000 RESET(H,K):RESET(H,L):SET(H+1,K):SET(H+2,L):RESET(H+1,K):RESET(H+2,L):SET(H
+1,J):SET(H+2,K):RESET(H+2,K):SET(H+1,J):SET(H+2,J):SET(H+3,J):P=H+4:W=J
1060 SET(P,W):RESET(P,W):P=P+1:SET(P,W)
1065 IFP<89THEN1060
1082 IF(POINT(P+1,W)=-1)AND(P<107)GOTO1200
1084 IF(POINT(P+1,W)=-1)AND(P>107)TT=TT+1:GOSUB8000
1090 IFP=125RESET(P,W):GOTO1110
1100 GOTO1060
```

```
1110 RESET(H+3,J):RESET(H+2,J):SET(H+2,K):RESET(H+2,K):SET(H+1,K):SET(H+2,L):RES
ET(H+1,J):RESET(H+1,K):RESET(H+2,L):SET(H,K):SET(H,L):RETURN
1200 RESET(P,W):P=P-1:SET(P,W):IFP>19GOTO1200
1205 IFPOINT(P-1,W)=-1GOTO1250
1240 GOTO1200
1250 RESET(H+1,J):RESET(H+2,J):RESET(H+3,J):RESET(H+4,J):RESET(A,K):RESET(A,L):S
ET(H,I):SET(H+1,I-1):SET(A-1,K):SET(A-2,L):RESET(H,I):RESET(H+1,I-1):RESET(A-1,K
):RESET(A-2,L):SET(H,I):SET(H,I-1):SET(A-1,J):SET(A-2,J):SET(A-3,J):RESET(A-1.J)

1260 RESET(A-2,J):RESET(A-3,J):SET(A,I):SET(A,I-1):SET(A,I-2):RESET(A,I-2):RESET
(A,J):RESET(A,I):RESET(A,I-1):RESET(D,I):RESET(E,I):RESET(H,I):RESET(H,I-1):SET(
G,I-1):SET(G,I):SET(H+1,I):SET(H+2,I):SET(H+4,I):SET(H+4,I-1):RESET(C.L)
1270 RESET(D,L):RESET(E,L):SET(G,L):SET(H,L):SET(H+1,L):RESET(C,K):RESET(D.K):RE
SET(E,K):RESET(F,K):SET(G,K):SET(H,K):SET(H+1,K):SET(H+2,K):RESET(C,J):RESET(D,J
):RESET(E,J):RESET(F,J):SET(H+1,J):SET(H+2,J):SET(H+3,J):SET(H+4,J):RESET(C,M)
1280 RESET(C,N):SET(D,N):SET(E,M):RESET(F,M):RESET(F,N):SET(H,N):SET(H+1,M):RESE
T(A,J):RESET(B,J):SET(B,N):SET(B,O):SET(C,O):SET(D,O):SET(E,O):SET(F.O):SET(G,O)
:SET(H,O):SET(G,N):SET(H,N):SET(H+1,N):SET(H,M):SET(H+1,M):SET(H+2,M):SET(H+1,L)

1290 SET(H+2,L):SET(H+3.L):SET(H+4,L):SET(H,K):SET(H+2,K):SET(H+3,K):SET(H+5,K):
SET(H,J):SET(H+6,J):RESET(G,I-1):RESET(G,I):RESET(H+1,I):RESET(H+2,I):RESET(H+4,
I):RESET(H+4,I-1):RESET(D,N):RESET(E,M):RESET(F,L):RESET(G,L):RESET(H,L)
1300 RESET(G,J):RESET(H+1,K):RESET(H+4,K):RESET(H+1,J):RESET(H+2.J):RESET(H+3.J)
:RESET(G,K):RESET(H+4,J):RESET(H,J):RESET(H,K):RESET(H+6,J):RESET(H+5,K):RESET(H
+2,K):RESET(H+3,K):RESET(H+1,L):RESET(H+2,L):RESET(H+3,L):RESET(H+4,L):RESET(H.M
)
1320 RESET(H+1,M):RESET(H+2,M):SET(H,O):SET(H+1,O):SET(H+2,O):SET(H+3,O):SET(H+4
,O):SET(H+5,O):SET(H+6,O):SET(H+7,O):SET(H+2,N):SET(H+3,N):SET(H+4,N):SET(H+5,N)
:FORX=1TO2000:NEXT:FORX=BTOH+7:RESET(X,N):RESET(X,O):NEXT
1330 SS=SS+1:IFSS=3THEN1400
1350 RETURN
1400 CLS:PRINTCHR$(23):PRINT"ALL YOUR MEN ARE DEAD":PRINT"    R. I. P. ":FORX=1TO
2000:NEXT:RUN
2000 POKE16070,186:POKE16072,181:POKE16006,171:POKE16007,191:POKE16008,151:POKE1
6009,149:POKE16005,170:POKE15943,176:RETURN
8000 RESET(P,W):RESET(P+1,W):P=125:IFTT=MMGOTO8020
8010 RETURN
8020 CLS:PRINTCHR$(23):PRINT"   YOU WIN !":FORX=1TO2000:NEXT:RUN
```

- 00000 -

** INVADERS **

```
1 ' COPYRIGHT (C) 1979
3 ' RONALD J. SULLY
4 ' 117 BRYANT RD., LOGANHOLME QLD 4129
5 CLEAR(60):DEFINTA-L,N-Y:DIME(80)
10 RANDOM:DEFSNGK:L1=132:SC=200:CA=4:CH=0:M1=30:MF=0:CLS:PRINTCHR$(23)
20 PRINT@22,"INVASION";
30 PRINT@86,STRING$(8,"*");
35 PRINT:PRINT:PRINT:PRINT:PRINT"A GAME OF CHANCE AND SKILL":PRINT:PRINT"A THRIL
L A SECOND ---
40 FORT=1TO69:E(T)=(510*RND(0))+15488:NEXT:FORT=1TO2900:NEXT
50 CLS:PRINT@26,"INSTRUCTIONS"
60 PRINT@90,STRING$(12,"-")
70 PRINT:PRINT"PRESS THE ARROW KEYS (";:PRINTCHR$(93);" ";CHR$(94);") TO MOVE LE
FT OR RIGHT."
```

```
80 PRINT"PRESS THE SPACE BAR TO FIRE THE LASER CANNON.
90 PRINT"PRESS THE (S) KEY TO START AGAIN.
100 T=502:PRINT:PRINT"1. FOR EACH HIT";:PRINT@T,"SCORE 100
110 T=T+64:PRINT"2. IF YOUR LASER CANNON IS DESTROYED";:PRINT@T,"LOSE  200
120 T=T+64:PRINT"3. EACH TIME YOUR CASTLES ARE HIT";:PRINT@T,"LOSE    5
130 T=T+64:PRINT"4. EACH TIME THE INVADERS MISS";:PRINT@T,"SCORE   5
140 T=T+64:PRINT"5. IF YOU MISS";:PRINT@T,"LOSE   10
150 T=T+64:PRINT"6. IF YOU HIT YOUR CASTLE";:PRINT@T,"LOSE   50
160 T=T+64:PRINT"7. IF YOUR 4 CANNONS ARE DESTROYED";:PRINT@T,"LOSE 1000
170 PRINT:PRINT"PRESS <<ENTER>>";
175 INPUTF$
180 CLS:POKEE(1),128:GOSUB1410:GOSUB1510:GOSUB1710
190 GOSUB1310:X=RND(2)
200 ONXGOSUB1310,1810
210 GOTO190
300 'SHOW SCORE
310 IFSC>HSTHENH5=SC
320 PRINT@0,"YOUR SCORE **";SC;" ** - HIGHEST SCORE **";HS;" **";
330 IFCA<0ORMF=83THEN370
340 FORT1=1TO79:IFPEEK(E(T1))<>32THEN360
350 GOSUB910:NEXT:GOTO370
360 RETURN
370 PRINT@48,STRING$(15." ");:PRINT@64,">>> END OF GAME <<<";:FORT=1TO3500:NEXT
GOTO10
899 END
900 'MOVE & FIRE
910 M1=M1-.05:IFM1<SM1=5
915 FORT=1TOM1:MF$=INKEY$:IFMF$=""THEN960
920 MF=ASC(MF$):IFMF=32GOSUB1210:GOTO970
930 IFMF=8GOSUB1010:GOTO970
940 IFMF=9GOSUB1110:GOTO970
950 IFMF=83GOTO370
960 NEXT
970 RETURN
1000 'LEFT MOVE
1010 IFPO-2<16321THEN1030ELSEPO=PO-2:POKEPO+1,32:POKEPO+2,32:POKEPO+3,32
1020 POKEPO-1,142:POKEPO,143:POKEPO+1,141
1030 RETURN
1100 'RIGHT MOVE
1110 PO=PO+2:IFPO>=16383THENPO=PO-2ELSEPOKEPO-2,32:POKEPO-3.32:POKEPO-1,32
1120 POKEPO-1,142:POKEPO,143:POKEPO+1,141
1130 RETURN
1200 'FIRE LASER
1210 FORRA=1TO14:EX=RA*64:CH=EX+64:POKEPO-EX,140:IFPEEK(PO-CH)<>32THEN1230
1220 POKEPO-EX,32:NEXT
1230 POKEPO-CH+64,32:FORT=1TO10:POKEPO-CH,42:POKEPO-CH,32:POKEPO-CH-1.42:POKEPO-
CH-1,32:POKEPO-CH+1,42:POKEPO-CH+1,32:NEXT:S=PO-CH
1250 IF5<15424SC=SC-10:GOTO1270
1255 IFS=>15424ANDS<16128SC=SC+100:GOTO1270
1260 IFS>16127SC=SC-50
1270 GOSUB310:RETURN
1300 'FIRE FROM INVASION FORCE
1310 R=RND(2):IFR=1THENL1=140:FORQ=79TO70STEP-1:GOTO1317
1315 FORQ=69TO1STEP-1
1317 IFPEEK(E(Q))=32THEN1370
1320 S=64:FORF=1TO16:K=E(Q)+S*F:IFK=>16383THENSC=SC+5:GOSUB310:GOTO1360
```

```
1330 IFPEEK(K)<>32THEN1380
1340 FORT=1TO2:POKEK,L1:POKEK,32:NEXT
1350 GOSUB910:NEXTF
1360 Z=Q:IFZ/2=INT(Z/2)GOSUB910
1365 GOSUB1810
1370 NEXTQ:L1=132:RETURN
1380 IFK>16127THENFORT=1TO20:POKEK,42:POKEK,32:NEXT
1385 IFK<16127THEN:POKEK,140:GOSUB310:GOTO1350
1390 IFK=>16128ANDK<16255THENSC=5C-5:GOSUB310:GOTO1360
1395 SC=SC-20:GOSUB310:GOSUB1410:RETURN
1400 'SET UP CANNON
1410 R=RND(3):PO=16350+R:FORCN=16320TO16383:POKECN,32:NEXT
1415 IFCA=05C=SC-1000:GOTO370
1420 CA=CA-1:POKEPO-1,142:POKEPO,143:POKEPO+1,141:SC=5C-200:GOSUB310
1425 IFCA=0THENFORT=1TO20:PRINT@48,STRING$(15," ");:PRINT@48,"* LAST CANNON *";:
NEXT
1430 RETURN
1500 'SET UP DEFENCE
1510 FORT=16196TO16251STEP10:FORN=T TO T+5:POKEN,143:NEXTN,T
1515 FORT=16132TO16187STEP10:FORN=T TO T+5:POKEN,184:NEXTN,T
1520 RETURN
1700 'SET UP INVASION FORCE
1710 FORT=1TO69:POKEE(T),140:NEXT
1720 C=63*RND(0)+129
1730 R=RND(2):FORT=70TO79
1740 K=T-70:E(T)=K*6+15427+R:POKEE(T)-1,139:POKEE(T),191:POKEE(T)+1,135:NEXT
1750 RETURN
1800 'MOVE INVASION FORCE
1810 FORM=1TO69:IFPEEK(E(M))=32THEN1860
1820 IFE(M)=>16380THENPOKEE(M),32:GOTO1850
1830 R=RND(2):POKEE(M),32:POKEE(M)+R,140
1840 E(M)=E(M)+R
1850 GOSUB910
1860 NEXT:RETURN
9999 ' !z1RESETRETURNz:RETURN
```

- 00000 -

** RANDOM NUMBER GENERATOR TEST PROGRAM 1 **

```
10 CLEAR 500:CLS
15 REM * PROGRAM BY KEITH NEIGHBOUR *
20 RANDOM
25 X$="
30 PRINT@128,"ENTER THE REQUIRED RANGE OF RANDOM NUMBERS FROM 1 TO ..";:INPUT E
32 PRINT@320,"IF A FIXED NUMBER OF R.N. SAMPLE5 ARE REQUIRED"
34 PRINT:PRINT"ENTER THE NUMBER ....... ELSE HIT ENTER TO RUN"
36 Q=0:INPUT Q
40 DIM C(E)
50 CLS:T=124:F=64:G=126
60 FOR N=1 TO E:GOSUB 500
70 PRINT@T+4*N,N;
80 NEXT N
82 FOR T=0 TO L:PRINT@T*128.">";:PRINT@T*128+60,"<";:NEXT T
85 C$="STOP OR START : ANY KEY (PRINT P, LIST H/L L)":PRINT@960,C$;
87 PRINT@942,"NO:TIMES:TOTAL";
```

```
90 R=RND(E):N=R:GOSUB 500:T=T+64
100 C(R)=C(R)+1:Z=C(R):K=K+1
102 PRINT@1005,"         ";
104 PRINT@1005,R;
106 PRINT@1009,Z;
108 PRINT@1014,K;
110 PRINT@T+4*R,Z;
112 PRINT@F," ";:PRINT@G," ";
113 F=(2*L+1)*64:G=F+60:PRINT@F,"*";:PRINT@G,"*";
115 IF K=Q GOTO 130
120 P$=INKEY$:IF P$="" GOTO 90
130 P$=INKEY$:IF P$="" GOTO 130
140 IF P$="L" GOSUB 1000:GOTO 170
150 IF P$="P" GOSUB 20000:GOTO 170
160 GOTO 90
170 IF INKEY$="" GOTO 170
180 PRINT@896,X$;:PRINT@960,C$;:GOTO 90
500 IF N>90 THEN T=404:L=6:RETURN
510 IF N>75 THEN T=336:L=5:RETURN
520 IF N>60 THEN T=268:L=4:RETURN
530 IF N>45 THEN T=200:L=3:RETURN
540 IF N>30 THEN T=132:L=2:RETURN
550 IF N>15 THEN T=64:L=1:RETURN
560 T=-4:L=0:RETURN
1000 PRINT@960,X$;:H=C(1)
1010 FOR N=2 TO E
1020 G=C(N)
1030 IF G>H THEN H=G
1040 NEXT N
1050 PRINT@896,"HIGHEST:";H;"FOR";
1060 FOR N=1 TO E
1070 J=C(N)
1080 IF J=H THEN PRINT N;
1090 NEXT N
1100 L=C(1)
1110 FOR N=2 TO E
1120 M=C(N)
1130 IF M<L THEN L=M
1140 NEXT N
1150 PRINT@960,"LOWEST :";L;"FOR";
1160 FOR N=1 TO E
1170 J=C(N)
1180 IF J=L THEN PRINT N;
1190 NEXT N:RETURN
20000 DEFINT A,S,B
20010 FOR S=15360 TO 16383 STEP 64
20020 FOR B=S+63 TO S+2 STEP -1
20030 IF PEEK(B)<>32 THEN 20050
20040 NEXT B
20050 A$="":FOR A=S TO B
20060 A$=A$+CHR$(PEEK(A))
20070 NEXT A
20080 LPRINT A$
20090 NEXT S
20100 RETURN
```

** RANDOM NUMBER GENERATOR TEST PROGRAM 2 **

```
10 CLS:DIM C(10):CLEAR 100
15 REM * PROGRAM BY KEITH NEIGHBOUR *
20 S$=STRING$(50,128)
30 RANDOM
40 PRINT@128,"IF A FIXED NUMBER OF R.N. SAMPLES ARE REQUIRED":PRINT
50 PRINT"ENTER THE NUMBER ....... ELSE HIT ENTER TO RUN"
60 A=0:INPUT A
70 GOSUB 240
80 P=1
90 R=RND(10)
100 C(R)=C(R)+1:K=K+1:Z=C(R)
110 L=842-R*64
120 Q=INT(Z/10)+1
130 IF Q>P THEN P=P+1:GOSUB 380
140 M=P*10
150 PRINT@954,M;
160 PRINT@995,K;
170 PRINT@ 832-R*64,Z;
180 N=INT(Z/P*5+.5)
190 PRINT@L,STRING$(N,140);
200 IF K=A GOTO 220
210 IF INKEY$="" GOTO 90
220 IF INKEY$="" GOTO 220
230 GOTO 90
240 CLS:PRINT@9,"F R E Q U E N C Y   O F   R A N D O M   N O S   1 - 10";
250 PRINT@73,"( HIT ANY KEY TO STOP OR START )";
260 PRINT@128," FR  NOS";
270 N=10:PRINT@196,N;CHR$(174);
280 N=9:FOR T=261 TO 773 STEP 64
290 PRINT@T,N;CHR$(174);
300 N=N-1
310 NEXT T
320 PRINT@840,CHR$(170);
330 FOR T=842 TO 892 STEP 5
340 PRINT@T,CHR$(176);
350 NEXT T
360 PRINT@969,"FREQUENCY FOR A TOTAL OF :";
370 RETURN
380 FOR U=1 TO 10
390 Y=C(U):N=INT(Y/P*5+.5):T=842-U*64
400 PRINT@T,S$;
410 PRINT@T,STRING$(N,140);
420 PRINT@T-10,Y;
430 NEXT U:RETURN
```

- OOOOO -

** ANALOGUE CLOCK USING "PRINT @" FOR GRAPHICS **

```
110 REM * THIS VERSION USES 'SET 2' AS PUBLISHED IN PREVIOUS EDITION OF MICRO 80
 *
120 REM * MEMORY SIZE = 32222 FOR 16K
130 DEFINTA-F,N-U
140 P=64:E=P:T=P:U=24:G=U:F=U:CLS:PRINT@128,"ENTER START TIME (H,M,S)";:INPUT H0
,M0,S0
```

```
150 CLS:PRINT@29,CHR$(188);CHR$(128);STRING$(2,140);CHR$(188)
160 PRINT@81,CHR$(188);CHR$(128);CHR$(188)
170 PRINT@93,CHR$(191);CHR$(128);CHR$(191);STRING$(2,179)
180 PRINT@108,CHR$(188)
190 PRINT@145,CHR$(191);CHR$(128);CHR$(191)
200 PRINT@172,CHR$(191)
210 PRINT@198,CHR$(176);CHR$(128);STRING$(3,176)
220 PRINT@244,STRING$(3,176)
230 PRINT@262,CHR$(191);CHR$(128);CHR$(191);CHR$(128);CHR$(191)
240 PRINT@308.CHR$(188);CHR$(140);CHR$(143)
250 PRINT@326,CHR$(131);CHR$(128);STRING$(3,131)
260 PRINT@372,STRING$(3,131)
270 PRINT@455,CHR$(191);CHR$(179);CHR$(191)
280 PRINT@501,CHR$(131);CHR$(179);CHR$(191)
290 PRINT@519,STRING$(2,140);CHR$(143)
300 PRINT@565,STRING$(2,140);CHR$(143)
310 PRINT@648.STRING$(3,176)
320 PRINT@692,CHR$(176);CHR$(128);CHR$(176)
330 PRINT@712,CHR$(191);CHR$(140);CHR$(191)
340 PRINT@756,CHR$(143);CHR$(140);CHR$(191)
350 PRINT@776,STRING$(3,131)
360 PRINT@822.CHR$(131)
370 PRINT@849,STRING$(2.131);CHR$(191)
380 PRINT@875,CHR$(191);STRING$(2.179)
390 PRINT@915,CHR$(143)
400 PRINT@926,CHR$(191);STRING$(2,179)
410 PRINT@939,STRING$(2,140);CHR$(143)
420 PRINT@990,CHR$(143);CHR$(140);CHR$(143);
430 PRINT@542,CHR$(130);CHR$(131);
440 PRINT@159,CHR$(140);
450 PRINT@233,CHR$(131);:PRINT@369,CHR$(131);
460 PRINT@563,CHR$(131);:PRINT@689.CHR$(176);
470 PRINT@809.CHR$(176);:PRINT@863.CHR$(140);
480 PRINT@789,CHR$(176);:PRINT@653,CHR$(176);
490 PRINT@523,CHR$(131);:PRINT@332,CHR$(131);
500 PRINT@213,CHR$(131);
510 M=H0*30+M0*.5:L=M0*6:K=S0*6
520 H1=37:V1=-15:A=62:B=24:H2=35:V2=-14:H3=29:V3=-10:M=M-.5:GOSUB610
530 K=K+6
540 FORQX=1TO515:NEXT:' ***** THIS TIMING LOOP AJUSTS ACCURACY
550 IFK=>360:K=K-360:L=L+6:GOSUB610
560 Z=K*.0174553:C=SIN(Z)*H1+A:D=COS(Z)*V1+B
570 RESET<E,F,A,B>
580 SET<A.B.C.D>
590 GOSUB670
600 E=C:F=D:GOTO530
610 IFL=>360:L=L-360:FORQA=1TO99:PRINT@0,CHR$(23);CHR$(28);:NEXT:K=K+6
620 M=M+.5:IFM=360:M=0
630 Z=L*.0174553:N=SIN(Z)*H2+A:O=COS(Z)*V2+B
640 Z=M*.0174553:R=SIN(Z)*H3+A:S=COS(Z)*V3+B
650 RESET<P,Q,A,B>
660 RESET<T.U,A,B>
665 K=K+1
670 SET<A,B,N,O>
690 SET<A.B,R,S>
700 P=N:Q=O:T=R:U=S
710 RETURN
```

***** BMON - PART 2 *****

THIS LISTING REPLACES THE LISTING IN LAST MONTH'S ISSUE.

```
7256  FD 21 FE 3F FD 7E 00 EE AF FD 77 00 C9 F3 31 15
7266  72 21 15 40 36 01 CD C9 01 21 7F 72 E5 21 90 7D
7276  7E 23 B7 C8 CD 33 00 18 F7 CD 49 00 FE 42 CA 5A
7286  7D FE 52 20 09 CD 45 73 CD F8 73 C3 63 72 FE 4C
7296  20 08 CD F8 7B CD 49 00 18 C3 FE 4D 20 18 CD C9
72A6  01 2A A4 40 23 23 ED 5B F9 40 B7 ED 52 D2 63 72
72B6  2A A4 40 C3 88 75 FE 50 20 06 CD 2A 75 C3 63 72
72C6  FE 56 CA 9A 78 FE 53 CA 57 75 FE 47 CA 76 76 FE
72D6  44 CA C1 76 FE 48 CA 1B 78 FE 46 20 06 CD 46 75
72E6  C3 63 72 FE 45 CA D5 75 FE 43 CA C1 79 C3 63 72
72F6  C1 79 C3 63 72 D5 11 00 00 DD 7E 00 D6 30 83 30
7306  01 14 5F C1 05 C8 C5 EB 29 E5 29 29 C1 09 EB DD
7316  23 18 E6 0E 05 06 0A C5 AF 0E 11 18 05 90 F2 2C
7326  73 80 29 17 18 03 29 17 23 0D 20 F1 1F C1 C6 30
7336  12 1B 0D 20 E2 C9 7E FE 31 D0 36 20 23 18 F7 2A
7346  A4 40 23 23 23 23 7E B7 CA E3 73 FD 21 4B 73 FD
7356  E5 FD 21 7E 73 FD E5 FE 8D C8 FE 9E C8 FE 91 C8
7366  FE 95 C8 FE 9D C8 FE 9F C8 FE B4 C8 FE B5 C8 FE
7376  B6 C8 FE CA C6 FD E1 C9 FD E1 23 E5 2B 23 7E B7
7386  28 0C FE 2C 28 08 FE 20 28 04 FE 3A 38 EF D1 E5
7396  B7 ED 52 7D 2A F9 40 B7 ED 52 13 E5 C1 03 2A F9
73A6  40 D5 E6 07 5F 3E 06 93 32 2A 72 85 6F D2 B7 73
73B6  24 E5 D1 2A F9 40 ED 53 F9 40 ED B8 D1 3A 2A 72
73C6  47 1B 3E 20 12 13 10 FC CD 08 7D E1 3A 2A 72 85
73D6  6F 30 01 24 7E FE 2C CA 80 73 C3 4C 73 CD 56 72
73E6  ED 5B F9 40 23 23 23 23 23 E5 B7 ED 52 E1 D0 C3
73F6  4C 73 DD E1 DD 22 3F 72 CD C9 01 3E 49 CD 33 00
7406  3E 3F CD 33 00 3E 20 CD 19 74 CD 50 74 ED 53 28
7416  72 18 71 CD 33 00 16 00 DD 21 2D 72 D5 CD 49 00
7426  CD 33 00 D1 FE 01 CA 63 72 FE 0D 28 10 FE 08 28
7436  08 DD 77 00 DD 23 14 18 E3 FD E1 18 BB 7A FE 06
7446  30 F7 DD 21 2D 72 CD FB 72 C9 ED 53 2B 72 16 00
7456  3E 4E CD 33 00 3E 3F CD 33 00 3E 20 CD 33 00 16
7466  00 DD 21 2D 72 D5 CD 49 00 CD 33 00 D1 FE 0D 29
7476  CC FE 01 CA 63 72 FE 08 20 02 18 CE DD 77 00 DD
7486  23 14 18 E1 2A A4 40 23 23 ED 4B F9 40 E5 B7 ED
7496  42 D2 86 77 E1 5E 23 56 ED 53 39 72 E5 EB 11 31
74A6  72 CD 19 73 21 2D 72 CD 3C 73 ED 5B 28 72 E1 72
74B6  2B 73 23 23 E5 EB 11 36 72 CD 19 73 21 32 72 CD
74C6  3C 73 2A A4 40 23 23 23 23 7E 23 B7 20 09 CD 56
74D6  72 7E B7 28 0D 18 EE 11 CF 74 D5 11 FB 74 D5 C3
74E6  5D 73 23 7E B7 CA 71 77 18 DC 23 10 FD 7E FE 2C
74F6  20 D7 23 18 02 FD E1 06 05 11 2D 72 7E 23 FE FF
7506  CA F0 74 7E 23 EB 4E EB 13 B9 20 DF 10 F5 01 06
7516  00 B7 ED 42 36 FF 23 11 32 72 06 05 1A 77 13 23
7526  10 FA 18 C9 2A A4 40 ED 5B F9 40 1B 1B B7 ED 52
7536  C8 19 22 3B 72 2A F9 40 36 00 2B 2B 22 A4 40 C9
7546  2A 3B 72 7C FE 42 D8 22 A4 40 21 00 00 22 3B 72
7556  C9 C3 63 72 00 11 04 00 19 7E 23 B7 20 FB EB E1
7566  73 23 72 EB 18 05 7E 23 B7 20 FB 7E 23 B7 20 F6
7576  7E 23 B7 20 F1 22 F9 40 22 FB 40 22 FD 40 C3 63
7586  72 EB 5E 23 56 23 1A B7 20 F7 13 1A 1B B7 20 F1
7596  5E 23 56 EB 01 0A 00 09 22 28 72 CD 2A 75 CD F8
75A6  7B 21 0A 00 22 2B 72 CD 45 73 DD 21 BB 75 DD 22
```

```
75B6  3F 72 C3 8A 74 CD 46 75 CD 49 00 C3 63 72 0D 48
75C6  45 58 2E 20 41 44 44 52 45 53 53 20 3F 20 00 CD
75D6  C9 01 21 C4 75 CD 76 72 CD 13 7A 3E 0D D5 CD 33
75E6  00 7C E6 F0 CD 4F 77 7C E6 0F CD 4F 77 7D E6 F0
75F6  CD 4F 77 7D E6 0F CD 4F 77 3E 20 CD 33 00 3E 3C
7606  CD 33 00 7E E6 F0 CD 4F 77 7E E6 0F CD 4F 77 3E
7616  3E CD 33 00 3E 20 CD 33 00 D1 06 02 11 3D 72 D5
7626  CD 49 00 D1 FE 01 CA 63 72 FE 08 28 AE FE 0A 20
7636  03 23 18 A7 FE 30 38 E7 FE 58 28 96 FE 5B 20 03
7646  2B 18 98 FE 47 30 D8 D5 CD 33 00 D1 12 13 10 CF
7656  11 3D 72 1A 13 D6 30 FE 10 38 02 D6 07 CD 9A 76
7666  4F 1A D6 30 FE 10 38 02 D6 07 81 77 23 C3 E1 75
7676  21 C4 75 CD 76 72 06 04 3E 20 CD 33 00 21 E6 41
7686  CD F9 79 06 02 21 E6 41 56 23 5E 7A 18 13 FE 3A
7696  30 0B D6 30 CB 27 CB 27 CB 27 CB 27 C9 D6 37 18
76A6  F3 CD 94 76 57 7B FE 3A 30 0D D6 30 82 57 23 D5
76B6  10 D6 E1 6C D1 62 E9 D6 37 18 F1 C3 63 72 16 00
76C6  21 32 72 3E 44 D5 CD 33 00 3E 3F CD 33 00 3E 20
76D6  CD 33 00 D1 D5 CD 49 00 D1 FE 0D 28 23 FE 08 20
76E6  03 D5 18 57 FE 3A 30 EC FE 01 CA 63 72 FE 30 38
76F6  E3 D5 CD 33 00 D1 77 23 14 7A FE 05 30 02 18 D4
7706  D5 3E 20 CD 33 00 3E 3D CD 33 00 3E 20 CD 33 00
7716  D1 7A B7 20 01 14 DD 21 32 72 CD FB 72 7A E6 F0
7726  CD 4F 77 7A E6 0F CD 4F 77 7B E6 F0 CD 4F 77 7B
7736  E6 0F CD 4F 77 D5 3E 48 CD 33 00 3E 0D CD 33 00
7746  D1 3E 30 32 32 72 C3 C4 76 FE 10 D4 68 77 FE 0A
7756  30 08 C6 30 D5 CD 33 00 D1 C9 C6 37 D5 CD 33 00
7766  D1 C9 CB 3F CB 3F CB 3F CB 3F C9 2A 28 72 ED 5B
7776  2B 72 19 22 28 72 E1 23 7E 23 B7 20 FB C3 8D 74
7786  E1 2A A4 40 CD 56 72 23 23 23 23 7E 23 E5 CD BD
7796  77 E1 B7 28 EF FE 2C 28 33 FE 22 28 0B 11 91 77
77A6  D5 11 D1 77 D5 C3 5D 73 7E 23 B7 28 D7 FE 22 28
77B6  DA FE 80 30 D6 18 F1 ED 4B F9 40 B7 ED 42 D8 E1
77C6  11 00 00 ED 53 2B 72 2A 3F 72 E9 D1 7E FE FF 28
77D6  06 FE 20 28 02 18 B4 E5 01 01 00 23 7E B7 28 08
77E6  FE 30 30 04 23 03 18 F4 D1 D5 C5 E5 D9 D1 C1 2A
77F6  F9 40 E5 B7 ED 42 22 F9 40 22 FB 40 22 FD 40 E1
7806  B7 ED 52 E5 D9 C1 ED B0 CD 08 7D E1 23 7E FE FF
7816  28 BA C3 91 77 C3 63 72 3E 48 CD 33 00 3E 3F CD
7826  33 00 3E 20 CD 33 00 21 3D 72 06 04 CD 49 00 FE
7836  08 28 5A FE 01 CA 63 72 FE 47 30 F0 FE 30 38 EC
7846  CD 33 00 77 23 10 E5 21 3D 72 06 02 7E 23 D6 30
7856  FE 0A 38 02 D6 07 CD 9A 76 57 7E 23 D6 30 FE 0A
7866  38 02 D6 07 82 57 D5 10 E3 E1 6C D1 62 3E 48 CD
7876  33 00 3E 20 CD 33 00 3E 3D CD 33 00 3E 20 CD 33
7886  00 ED 5B 20 40 06 05 13 10 FD CD 19 73 3E 0D CD
7896  33 00 18 84 C3 63 72 06 1A 0E 41 2A A4 40 CD 2B
78A6  00 FE 01 CA 63 72 23 23 23 23 7E 23 ED 5B F9 40
78B6  E5 ED 52 30 1E E1 E5 DD E1 DD 2B DD 2B B7 28 E6
78C6  FE 22 28 1C FE 93 CA 73 79 FE FB CA 73 79 B9 28
78D6  2A 18 D7 E1 0C 10 C4 CD 56 72 CD 49 00 C3 63 72
78E6  7E 23 B7 28 C1 FE 22 28 06 00 00 00 00 18 F1 7E
78F6  23 B7 28 B2 FE 3B 28 B2 C3 B2 78 FE 20 28 AB 57
7906  DD 7E 00 FE 5B 30 06 FE 41 38 02 18 9D 7E 23 B7
7916  28 33 FE 3A 38 08 FE 41 30 04 1E 20 18 11 FE 5B
7926  30 F8 FE 2C 28 F4 FE 28 28 F0 FE 29 28 EC 5F E5
7936  C5 21 00 3C 01 FF 03 7A ED B1 E2 4E 79 7E BB 20
7946  F6 C1 E1 18 16 2B 18 D2 C1 E1 7A D5 CD 33 00 D1
```

```
7956  7B D5 CD 33 00 3E 2C CD 33 00 D1 7E 23 B7 CA AC
7966  78 FE 3A CA B0 79 FE 5B 38 F1 C3 B2 78 7E 23 B7
7976  20 FB C3 AC 78 53 54 41 52 54 20 45 4E 44 20 20
7986  45 4E 54 52 59 20 4E 41 4D 45 20 3F 0D 00 54 59
7996  50 45 20 3C 20 42 52 45 41 4B 20 3E 20 54 4F 20
79A6  53 54 4F 50 20 44 55 52 49 4E 47 20 43 41 53 53
79B6  45 54 54 45 20 49 2F 4F 2E 0D 00 CD C9 01 21 7B
79C6  79 CD 76 72 18 57 CD 49 00 FE 01 CA 63 72 FE 08
79D6  28 0E FE 30 38 F0 FE 47 30 EC CD 33 00 77 23 C9
79E6  5F 78 FE 04 7B 30 DF 2B 04 CD 33 00 18 D8 06 04
79F6  21 32 72 CD CC 79 10 FB C9 7E 23 CD 94 76 4F 7E
7A06  23 FE 3A 30 04 D6 30 18 02 D6 37 81 C9 CD F4 79
7A16  21 32 72 CD FF 79 57 CD FF 79 5F EB C9 CD 13 7A
7A26  22 3F 72 3E 20 CD 33 00 CD 33 00 CD 13 7A 22 41
7A36  72 3E 20 CD 33 00 CD 13 7A 22 43 72 3E 20 CD 33
7A46  00 CD 33 00 36 06 21 45 72 CD 49 00 FE 01 CA 63
7A56  72 FE 0D 28 0D FE 08 28 32 77 23 CD 33 00 10 E9
7A66  18 06 3E 20 77 23 10 FC 3E 0D CD 33 00 21 94 79
7A76  CD 76 72 2A 41 72 ED 5B 3F 72 B7 ED 52 23 11 80
7A86  00 06 00 B7 ED 52 38 11 04 18 F8 5F 78 FE 06 7B
7A96  30 B7 CD 33 00 2B 04 18 B0 19 7D 32 26 72 21 45
7AA6  72 AF C5 CD 12 02 CD 87 02 C1 3E 55 CD 64 02 C5
7AB6  06 06 7E 23 CD 64 02 10 F9 2A 3F 72 C1 78 B7 28
7AC6  3B CD 56 72 3E 3C CD 64 02 7B CD 64 02 7D 57 CD
7AD6  64 02 7C CD 64 02 82 57 3A 40 38 E6 04 20 2A 7E
7AE6  23 CD 64 02 1D 20 EF 82 CD 64 02 1E 80 16 00 10
7AF6  D0 E5 ED 5B 41 72 B7 ED 52 30 0E E1 16 00 3A 26
7B06  72 06 01 B7 28 03 5F 18 B8 2A 43 72 3E 78 CD 64
7B16  02 7D CD 64 02 7C CD 64 02 CD F8 01 C3 63 72 42
7B26  41 53 49 43 20 4D 4F 4E 49 54 4F 52 20 42 59 20
7B36  45 2E 52 2E 50 41 41 59 20 28 43 29 20 31 39 37
7B46  39 20 28 48 49 54 20 3C 53 48 49 46 54 3E 20 5C
7B56  20 54 4F 20 41 43 43 45 53 53 2E 29 42 41 53 49
7B66  43 20 50 52 4F 47 52 41 4D 20 4C 4F 43 41 54 49
7B76  4F 4E 53 20 3A 20 53 54 41 52 54 20 3D 20 58 58
7B86  58 58 48 20 2C 20 45 4E 44 20 3D 20 58 58 58 58
7B96  48 20 20 CD 5F 7B C3 CC 06 CD C9 01 21 40 3C 22
7BA6  20 40 11 00 3C 21 25 7B 01 3D 00 ED B0 21 C6 7B
7BB6  3E C3 32 33 40 22 34 40 AF 32 15 40 21 99 7B C9
7BC6  CD E3 03 B7 C8 FE 1A C2 85 7D ED 73 24 72 31 22
7BD6  72 E5 21 00 00 22 2B 72 D5 C5 F5 DD E5 FD E5 C3
7BE6  63 72 52 45 41 44 59 20 43 41 53 53 45 54 54 45
7BF5  00 0D CD C9 01 01 00 50 CD 60 00 21 E8 7B CD 76
7C06  72 01 00 50 CD 60 00 CD 2B 00 B7 28 E5 FE 0D C2
7C16  63 72 CD C9 01 21 94 79 CD 76 72 21 FA 3F 22 20
7C26  40 2A A4 40 E5 DD E1 CD C4 7C 3E 00 CD 12 02 CD
7C36  96 02 CD 35 02 FE D3 28 F9 21 40 3C 06 04 CD 35
7C46  02 DD 77 00 DD 23 10 F6 3A 40 38 E6 04 20 45 CD
7C56  35 02 DD 77 00 DD 23 B7 28 15 FE 80 30 47 77 23
7C66  7C FE 3F 38 E3 7D FE B7 38 DE 21 40 3C 18 D9 36
7C76  AA 23 36 BF CD 35 02 DD 77 00 DD 23 DD E5 D1 CD
7C86  71 7D 06 03 B7 20 B7 CD 35 02 DD 77 00 DD 23 06
7C96  02 B7 20 AA DD 22 F9 40 DD 22 FB 40 DD 22 FD 40
7CA6  CD F8 01 18 36 E5 D6 80 87 26 7F 6F 5E 23 56 E1
7CB6  1A E6 7F 77 23 13 1A FE 5E 30 8D 77 18 F6 01 D0
7CC6  01 21 50 16 11 00 7F 7E E6 80 C4 DA 7C 23 0B 78
7CD6  81 C8 18 F3 EB 73 23 72 23 EB C9 DD 2B DD 22 26
7CE6  72 21 62 7B 11 C0 3F 01 37 00 ED B0 2A A4 40 11
```

```
7CF6  E2 3F CD 26 7D 2A 26 72 11 F0 3F CD 26 7D CD 08
7D06  7D C9 2A A4 40 E5 23 23 23 23 7E B7 20 FB 23 D1
7D16  EB 73 23 72 EB 7E B7 20 EC 23 7E B7 C8 2B 18 E5
7D26  7C CB 3F CB 3F CB 3F CB 3F CD 4B 7D 7C E6 0F CD
7D36  4B 7D 7D CB 3F CB 3F CB 3F CB 3F CD 4B 7D 7D E6
7D46  0F CD 4B 7D C9 FE 0A 38 06 D6 09 F6 40 18 02 F6
7D56  30 12 13 C9 CD C9 01 21 15 40 36 00 FD E1 DD E1
7D66  F1 C1 D1 E1 ED 7B 24 72 C3 CC 06 E5 21 EA 71 ED
7D76  52 E1 D0 CD 46 75 AF 32 15 40 3E 07 C3 F4 1F C5
7D86  01 0F 00 10 FE 0D 20 FB C1 C9 42 41 53 49 43 20
7D96  4D 4F 4E 49 54 4F 52 20 56 45 52 2E 20 31 2E 32
7DA6  20 20 43 4F 4D 4D 41 4E 44 20 20 4C 49 53 54 20
7DB6  3A 0D 2D 2D 2D 20 53 45 4C 45 43 54 20 46 55 4E
7DC6  43 54 49 4F 4E 20 21 21 21 20 2D 2D 2D 0D 42 20
7DD6  2D 20 52 45 54 55 52 4E 20 54 4F 20 42 41 53 49
7DE6  43 0D 52 20 2D 20 52 45 4E 55 4D 42 45 52 0D 4C
7DF6  20 2D 20 4C 4F 41 44 20 42 41 53 49 43 20 50 52
7E06  4F 47 52 41 4D 0D 4D 20 2D 20 4D 45 52 47 45 20
7E16  42 41 53 49 43 20 50 52 4F 47 52 41 4D 0D 50 20
7E2S  2D 20 50 52 4F 54 45 43 54 20 50 52 4F 47 52 41
7E36  4D 0D 56 20 2D 20 4C 49 53 54 20 56 41 52 49 41
7E46  42 4C 45 53 20 55 53 45 44 20 49 4E 20 50 52 4F
7E56  47 52 41 4D 0D 53 20 2D 20 52 45 53 54 4F 52 45
7E66  20 50 52 4F 47 52 41 4D 0D 47 20 2D 20 47 4F 54
7E76  4F 20 48 45 58 20 4C 4F 43 41 54 49 4F 4E 0D 44
7E86  20 2D 20 44 45 43 2E 20 54 4F 20 48 45 58 2E 20
7E96  43 4F 4E 56 45 52 53 49 4F 4E 0D 48 20 2D 20 48
7EA6  45 58 2E 20 54 4F 20 44 45 43 2E 20 43 4F 4E 56
7EB6  45 52 53 49 4F 4E 0D 46 20 2D 20 43 41 4E 43 45
7EC6  4C 20 50 52 4F 54 45 43 54 49 4F 4E 0D 45 20 2D
7ED6  20 45 44 49 54 20 4D 45 4D 4F 52 59 0D 43 20 2D
7EE6  20 43 4F 50 59 20 4D 45 4D 4F 52 59 20 54 4F 20
7EF6  54 41 50 45 0D 00 00 00 00 04 50 15 53 16 56 16
```

- 00000 -

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

MICRO-80 PRODUCTS MICRO-80 PRODUCTS MICRO-80 PRODUCTS MICRO-80 PRODUCTS

WE are delighted to be able to announce the release of

ESCAPE FROM THE DALEKS

A top quality real-time Level 2 game, written by none other than our own Peter Hartley.

ESCAPE FROM THE DALEKS offers you all the excitement and thrills of helping Dr Who to escape from a whole mess of trouble with his eternal enemies. This game runs in real time, offers continuously variable levels of play from Easy to Utterly Impossible, and comes ready to load, on cassette, complete with insults and mediocre messages of praise. Guaranteed to offer hours of fun for young and old alike, as the game levels are automatically adjusted after each game, ensuring that the player is always going to have to work very hard to win!

The regular price for this program will be $15.00, however, as a SPECIAL OFFER to subscribers, all orders postmarked APRIL 21st or earlier. will be processed at $12.50 (which saves you the cost of this edition of MICRO-80).

So DON'T DELAY - order your copy today. You'll need all the practice you can get if you're going to beat your mates!
Level 2/16K

- -

********** SPECIAL OFFER **********

BMON ON CASSETTE ----------- ONLY $10.00 (SAVE $5.00)

This is the Utility program featured in the February, March and April issues. The complete program is available NOW, on cassette. Each cassette comes complete with full documentation and contains the 16K, 32K and 48K versions.

HARD COPY LISTING of the ASSEMBLED SOURCE CODE ---- ONLY $5.00

Order before April 21st and obtain these special prices.
Level 2/16K

- -

********** ANNOUNCING TOUCHTYPE ********

Learn to touch type on your '80. Cassette with 20 lessons - yes that's right 20 LESSONS -------------- ONLY $19.95

TOUCHTYPE takes you from a raw beginner to a competent touch typist, easily and enjoyably. Speed up your program entry. Learn to input through the keyboard without looking at the keys.
Level 2/4K.

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

********** SUBSCRIPTOON APPLICATION **********

To: MICRO-80, PO BOX 213, GOODWOOD, SA 5034. Phone:(08) 381 8542

Please send me:

- CURRENT ISSUE	$2.50
- CASSETTE - SINGLE ISSUE..................	$3.50
- 12 MONTH SUBSCRIPTION, MAGAZINE ONLY	$24.00
- 12 MONTH SUBSCRIPTION, MAGAZINE PLUS CASSETTE .	$60.00

(Make cheques payable to MICRO-80)

NAME:...

ADDRESS:..

....

....POST CODE. ....

* *

MICRO-80 PRODUCTS MICRO-80 PRODUCTS MICRO-80 PRODUCTS MICRO-80 PRODUCTS

********** ORDER FORM **********

To MICRO-80 PRODUCTS, P.O. BOX 213, GOODWOOD, SA 5034

Please send me:-

LEARNING LEVEL II............................	$16.95
TOUCHTYPE....................................	$19.95
BMON - CASSETTE..............................	$10.00
BMON - SOURCE LISTING........................	$ 5.00
RPN CALCULATOR...............................	$19.95
ESCAPE FROM THE DALEKS.......................	$12.50

NOTE: Add $0.50 postage and packing for each program, $1.20 for Learning Level II.

Please make cheques and Money Orders payable to MICRO-80.

NAME:..

ADDRESS:...

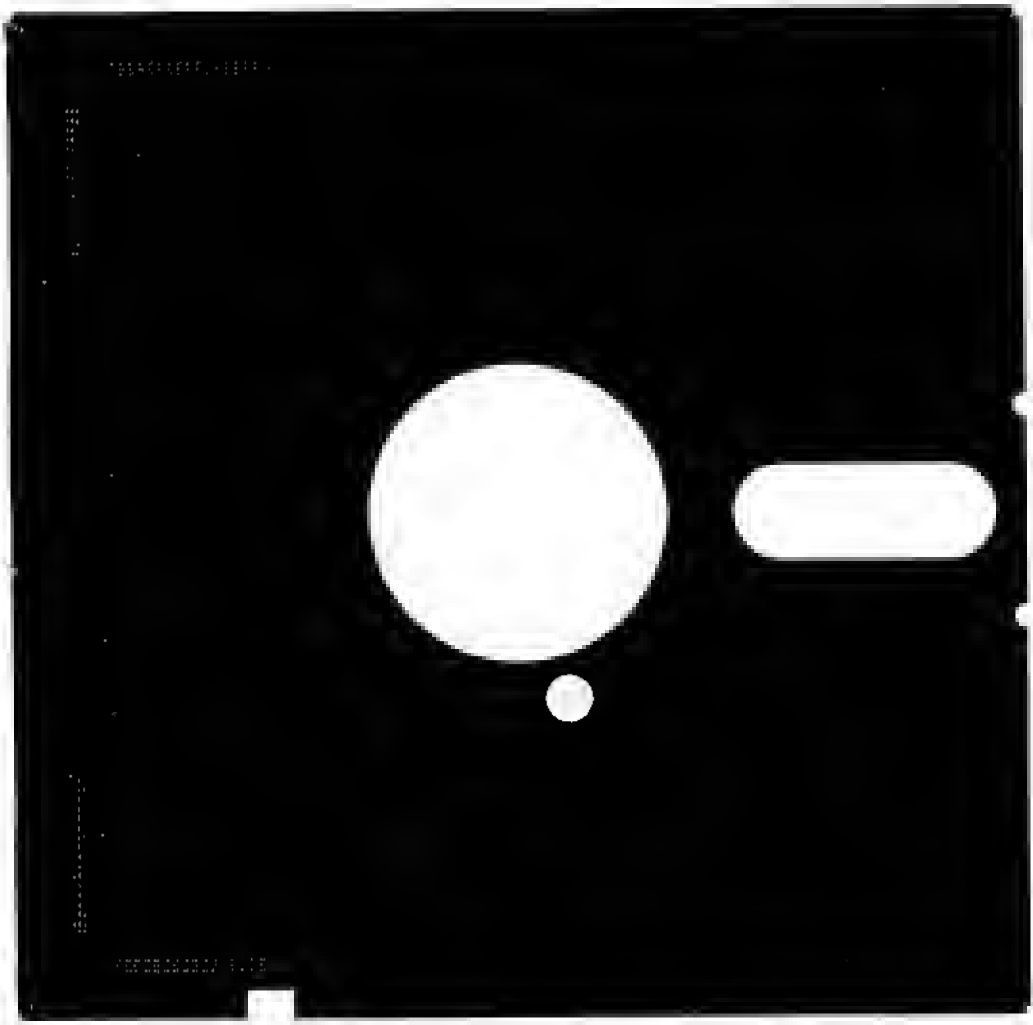
....

....................................POST CODE............

* *

MICRO-80

P.O. 8OX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

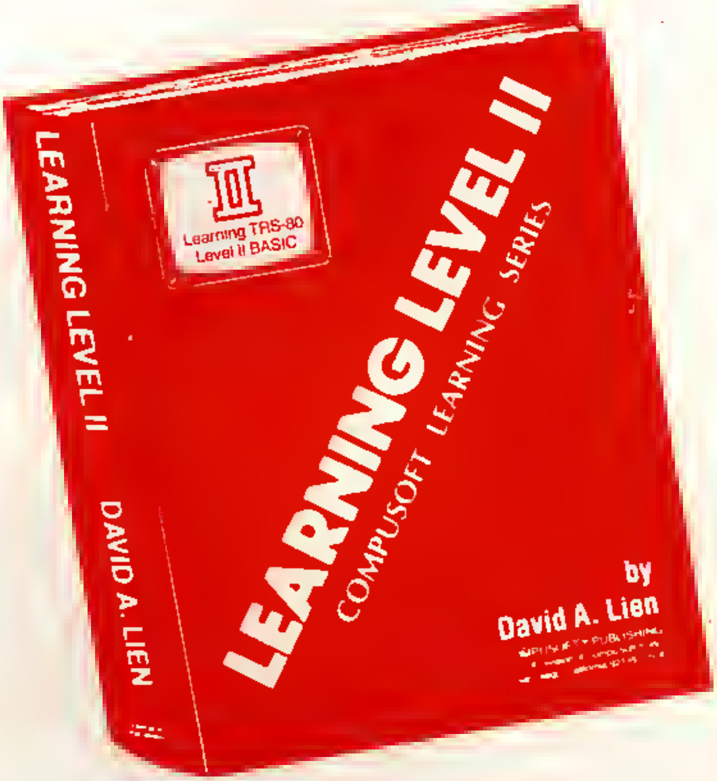


TEMPLATE - see page 15

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

It's Here At Last . . .



Written by the author of your Level I Users Manual, LEARNING LEVEL II picks right up where the Level I Manual leaves off. It also supplies the changes needed to make the Level I Manual compatible with your Level II TRS-80.

LEARNING LEVEL II covers all Level II BASIC beyond Level I, plus much more. It shows you how to use the Editor, explains what the many error messages are really saying, and leads you thru conversions of Level I programs to Level II.

Oual cassettes, printers, the Expansion Interface with clock and other features are explained in the same easy-to-learn style that made the Level I Manual famous. LEARNING LEVEL II was created specifically for your Level II TRS-80!

MICRO-80